Complex & Intelligent Systems (2023) 9:6697–6728
https://doi.org/10.1007/s40747-023-01105-4

ORIGINAL ARTICLE

Multifactorial evolutionary algorithm with adaptive transfer strategy based on decision tree

Wei Li[1] · Xinyu Gao[1] · Lei Wang[2]

Received: 30 January 2023 / Accepted: 8 May 2023 / Published online: 29 May 2023

Abstract
Multifactorial optimization (MFO) is a kind of optimization problem that has attracted considerable attention in recent years. The multifactorial evolutionary algorithm utilizes the implicit genetic transfer mechanism characterized by knowledge transfer to conduct evolutionary multitasking simultaneously. Therefore, the effectiveness of knowledge transfer significantly affects the performance of the algorithm. To achieve positive knowledge transfer, this paper proposed an evolutionary multitasking optimization algorithm with adaptive transfer strategy based on the decision tree (EMT-ADT). To evaluate the useful knowledge contained in the transferred individuals, this paper defines an evaluation indicator to quantify the transfer ability of each individual. Furthermore, a decision tree is constructed to predict the transfer ability of transferred individuals. Based on the prediction results, promising positive-transferred individuals are selected to transfer knowledge, which can effectively improve the performance of the algorithm. Finally, CEC2017 MFO benchmark problems, WCCI20-MTSO and WCCI20-MaTSO benchmark problems are used to verify the performance of the proposed algorithm EMT-ADT. Experimental results demonstrate the competiveness of EMT-ADT compared with some state-of-the-art algorithms.

Keywords Multifactorial optimization · Evolutionary algorithm · Knowledge transfer · Decision tree

Introduction

In recent years, many researchers show great interest on a new category of optimization problems, which is called multifactorial optimization (MFO). Different from well-known optimization problems, such as single objective optimization and multiobjective optimization, MFO aims to handle multiple distinct optimization tasks in one run. Such problems widely exist in the fields of science, engineering and technology. For example, in a complex supply chain problem [1], two optimization problems, i.e. shop scheduling (production optimization) and vehicle routing (logistics optimization), are involved simultaneously. Gupta et al. [1] was the first to use evolutionary algorithm to deal with single objective MOP problems, which is called multifactorial evolutionary algorithm (MFEA). Subsequently, a series of multifactorial evolutionary algorithms are proposed and applied to deal with different optimization problems, such as job shop scheduling [2], shortest-path tree [3], ensemble classification [4] and multiobjective optimization [5].

MFEA is characterized by integrating cultural effects through assortative mating and vertical cultural transmission. In MFEA, a prescribed parameter called random mating probability (*rmp*) is used to control knowledge transfer during the optimization process. Due to lack of prior knowledge about intertask similarity, for related multiple optimization tasks, *rmp* can enhance the optimization performance, which is called positive transfer; on the contrary, the optimization performance may deteriorate for unrelated multiple optimization tasks, which is called negative transfer. Since knowledge transfer is very important to multifactorial evolutionary algorithms, researchers have developed many transfer strategies to alleviate negative knowledge transfer between unrelated tasks. These strategies can be divided into four categories:

1. *Domain adaptation techniques*: Bali et al. [6] proposed a linearized domain adaptation (LDA) strategy, which

✉ Wei Li
liwei@xaut.edu.cn

1 School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China

2 Shaanxi Key Laboratory for Network Computing and Security Technology, Xi'an 710048, China

transforms the search space to improve the correlation between representative space and its constitutive task. Accordingly, an effective platform is provided for knowledge transfer. Wang et al. [7] developed an explicit autoencoding strategy to incorporate multiple search mechanisms with different biases in the evolutionary multitasking paradigm. The proposed autoencoder aims to learn a mapping between problem domains. Xue et al. [8] proposed an affine transformation-enhanced MFO algorithm (AT-MFEA) to enhance the transferability between distinct tasks. In AT-MFEA, a superior intertask mapping is obtained by the rank loss function. Moreover, the evolutionary-path-based representation model is established to bridge the gap between two distinct problems from different domains.

2. *Adaptive strategy of parameter rmp*: Ong et al. [9] proposed an online transfer parameter estimation strategy (MFEA-II) to minimize damage caused by negative transfer between tasks. In MFEA-II, the parameter *rmp* takes the form of a symmetric matrix instead of a scalar value. The constructed matrix can capture the non-uniform inter-task synergies even if the complementarity between tasks may not be uniform across different task-pairs. Moreover, the RMP matrix is continuously learned and adapted during the search process, which is helpful to obtain the global optimums of different tasks. Xu et al. [5] proposed a cultural transmission based multi-objective evolution strategy (CT-EMT-MOES), where an adaptive information transfer strategy is developed to adaptively adjust the parameter *rmp*. In detail, the proposed transfer strategy utilizes the mutation success rate of the target itself and the success rate of the information transfer to reasonably allocate the evolution resources between tasks. Li et al. [10] proposed an explicit multipopulation evolutionary framework (MPEF) to improve information transfer effects. In MPEF, the parameter *rmp* is adjusted based on the evolution status of the population. Specifically, if the ratio that offspring is superior to its parent is larger than the given threshold, the *rmp* will not be adjusted because the population evolves well with the current *rmp*. On the contrary, the *rmp* will be updated because the current *rmp* causes a negative transfer.
3. *Intertask learning strategy*: Da et al. [11] proposed a transfer evolutionary computation paradigm (AMTEA), which can reduce the risks of negative transfer via online source-target similarity learning. In AMTEA, a probabilistic model is constructed with the distribution of elite solutions from some source optimization task. Subsequently, the probabilistic model is used to provide a promising direction for the search on a related target task. Gao et al. [12] utilized semi-supervised learning strategy to enhance the effectiveness of knowledge transfer (EMT-SSC). In EMT-SSC, the promising individuals are identified with semi-supervised learning strategy. Then, these individuals transfer valuable knowledge between tasks. Zheng et al. [13] developed a self-regulated evolutionary multitask optimization (SREMTO) algorithm to dynamically adjust the intensity of knowledge transfer between tasks. In SREMTO, a task group is created based on the ability vectors of individuals. The degree of overlap between the task groups depends on the degree of tasks relatedness. The cross-task knowledge transfer is conducted through the overlapping parts between task groups.
4. *Multi-knowledge transfer mechanism*: Cai et al. [14] proposed a hybrid knowledge transfer strategy to conduct information transfer between tasks (EMTO-HKT). In EMTO-HKT, a multi-knowledge transfer mechanism including an individual-level learning strategy and a population-level learning strategy are used to transfer knowledge according to the degree of the task relatedness. Liang et al. [15] proposed a two-stage adaptive knowledge transfer mechanism. At the first stage, the search step of each individual is adjusted to alleviate the negative transfer, while at the second stage, the search range of each individual is adjusted to improve the exploration ability of the population. Ding et al. [16] proposed a generalized multitasking evolutionary optimization for expensive problems (G-MFEA). In G-MFEA, two strategies are proposed to conduct knowledge transfer between optimization problems with different locations of the optimums and different numbers of decision variables.

Although existing MFEAs endeavor to alleviate negative transfer during the optimization process, the solution precision obtained by these algorithms is not satisfactory, especially for those multitasking problems with low relatedness. Further, individuals with useless knowledge for other tasks are often transferred due to the lack of prior information about the relatedness between tasks, which obviously results in a waste of resources. To solve these problems, this paper presents an evolutionary multitasking optimization algorithm with adaptive transfer strategy based on the decision tree (EMT-ADT). In EMT-ADT, the decision tree based on Gini coefficient is constructed to predict the individual transfer ability. To the best of our knowledge, this is the first attempt in the literature to use the decision tree to enhance positive knowledge transfer in the MFO paradigm. The primary contributions of this paper can be summarized as follows:

1. The transfer ability of individuals is defined to quantify the amount of useful knowledge contained in the transferred individuals. Individuals with high transfer ability are used to construct a decision tree.

2. Combine with a knowledge of supervised machine learning, the proposed algorithm uses decision tree to predict the positive-transferred individuals. By selecting promising positive-transferred individuals, the proposed algorithm can improve the probability of a positive transfer.
3. The success-history based adaptive differential evolution algorithm (SHADE) is used as the search engine, which can demonstrate the generality of the MFO paradigm. Three multifactorial optimization benchmark sets are used to verify the competitiveness of the proposed method.

The rest of this paper is organized as follows: the next section introduces the details of MFO and decision tree model. The next section describes the proposed EMT-ADT algorithm. The following section presents the experimental results on three multifactorial optimization (MFO) benchmark sets and two combinatorial optimization problems (TSP and TRP). The conclusion and future work are summarized in the last section.

MFO and decision tree

Multifactorial optimization

As mentioned before, multifactorial optimization (MFO) is an evolutionary multitasking paradigm that aims to find a group of optimal solutions simultaneously, each of which corresponds to an optimization problem. To compare individuals in a multitasking environment conveniently, it is necessary to encode and decode different individuals in a unified search space. For an unconstrained multitasking problem with n tasks, T_j denotes the jth task with a search space X_j and an objective function f_j. For the ith individual p_i, its properties are defined as follows [1]:

Definition 1 The factorial cost Ψ_j^i of individual p_i on task T_j is the objective value f_j^i of individual p_i.

Definition 2 The factorial rank r_j^i of individual p_i on task T_j is the index of p_i, provided that the population is sorted in ascending order according to Ψ_j.

Definition 3 The scalar fitness of individual p_i is defined as $'_i = 1/\min_{j\epsilon\{1,\ldots,n\}}\{r_j^i\}$.

Definition 4 The skill factor τ_i of individual p_i is defined as $\tau_i = \text{argmin}_{j\epsilon\{1,\ldots,n\}}\{r_j^i\}$. In other words, τ_i denotes the index of the task that individual p_i performs the best among all other tasks.

MFEA is a pioneer evolutionary algorithm to realize the MFO paradigm, which transfers genes and memes through assortative mating and vertical cultural operation. Algorithm 1 presents the basic framework of MFEA.

Algorithm 1. MFEA Framework

```
1:  Initialize the population P
2:  Evaluate the population on task $T_i$, $i = 1,\ldots,m$
3:  Compute $\tau_i$, $i = 1,\ldots,N$
4:  while the termination criteria are not met do
5:      Generate the offspring S by assortative mating
6:      Evaluate the offspring S of the selected optimization tasks
7:      Form an intermediate population U = P ∪ S
8:      Update $\varphi_i$ and $\tau_i$ of intermediate population U
9:      Select the fittest individuals from U to form the next population P
10: end while
```

Decision tree

The decision tree is one of the supervised machine learning method to multistage decision making [17]. A tree structure is used to present the decision rules summarized from a series of data with characteristics and labels. Generally, a decision tree consists of a root node, some internal nodes and leaf nodes. Leaf nodes are nodes that have no appropriate descendants, while other nodes (except the root) are called internal nodes. Each internal node is associated with a test attribute, each branch represents the outcome of the test, while each leaf node assigns one or more class labels [18, 19]. A path from the root node to each leaf node corresponds to a decision test sequence.

The design of a decision tree mainly includes three tasks [17].

Task 1: Select a node splitting rule.

Table 1 An instance of playing training dataset

ID	a_1	a_2	a_3	a_4	L
1	Sunny	Hot	High	F	N
2	Sunny	Hot	High	T	N
3	Overcast	Hot	High	F	Y
4	Rainy	Mild	High	F	Y
5	Rainy	Cool	Normal	F	Y
6	Rainy	Cool	Normal	T	N
7	Overcast	Cool	Normal	T	Y
8	Sunny	Mild	High	F	N
9	Sunny	Cool	Normal	F	Y
10	Rainy	Mild	Normal	F	Y
11	Sunny	Mild	Normal	F	Y
12	Overcast	Mild	High	T	Y
13	Overcast	Hot	Normal	F	Y
14	Rainy	Mild	High	T	N

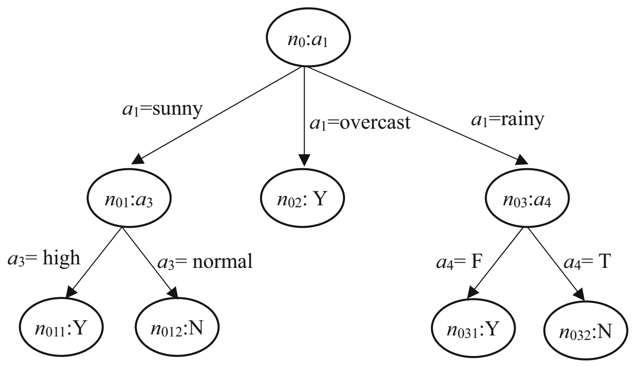


Fig. 1 Decision tree trained by Table 1

Task 2: Decide which nodes are terminal.

Task 3: Assign each terminal node to a class label.

Taking the playing dataset as an example, Table 1 shows an instance of playing training dataset, which has fourteen samples. Each sample consists of ID, four attributes and one category label. The attribute consists of weather (a_1), temperature (a_2), humidity (a_3) and whether there is wind (a_4). The category label (L) is whether to play today.

Figure 1 shows the decision tree trained by Table 1. n_i represents the ith node.

Proposed method

Motivations

Positive knowledge transfer has a significant effect on the performance of multifactorial evolutionary algorithm. In MFEA, the knowledge transfer among tasks is controlled by the parameter *rmp*. In each task, since the solutions are randomly selected to exchange information based on the same probability, there is a possibility that the transfer turns out to be negative, thereby leading to deterioration of algorithm performance [20, 21]. Therefore, in the case of uncertain correlation between tasks, how to accurately select valuable solutions is core to improve the performance of the algorithm. To solve these problems, this paper proposed an evaluation rule to quantify the transfer ability of individuals involved in knowledge transfer.

Furthermore, the research shows that the decision tree model has the following advantages in the construction process and data processing. First of all, the decision tree uses the knowledge learned from training to directly form a hierarchical structure, which is readable and easy to understand and implement. Secondly, the decision tree is suitable for data sets with small size, and the time complexity of the decision tree algorithm is small. Finally, the decision tree is not sensitive to missing values during data processing. In addition, the decision tree can deal with irrelevant feature data, and can accurately predict the results of analysis with large data sources in a relatively short time. Therefore, in the proposed EMT-ADT algorithm, the decision tree is constructed according to the information of transferred individuals. The number of individuals for each knowledge transfer is set to 10, and five consecutive generations of transferred individuals are used as the training data to construct the decision tree model, which conforms to the characteristics of the decision tree. Based on the constructed decision tree, promising positive-transferred individuals are selected to conduct knowledge transfer between tasks, which can achieve a fast convergence and improve the solution accuracy.

Definition of transfer ability

For a population $P = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, an archive $A = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N\}$ is used to store individuals involved in knowledge transfer; the historical transferred population $TP = \{\mathbf{t}_i | 1 \le i \le n \wedge \omega(\mathbf{t}_i) > \omega(\mathbf{t}_{n+1})\}$ is used to store n individuals with the highest knowledge transfer ability in the archive A. Each $\mathbf{t}_i$ has an associated subset Φ_i.

For each $\mathbf{t}_i \in TP$, if $\mathbf{t}_i$ participates in the generation of offspring $\mathbf{y}_j$ (recorded as $\mathbf{t}_i \rightarrow \mathbf{y}_j$), the associated subset Φ_i is defined as follows.

$$\Phi_i = \{\mathbf{y}_j | \mathbf{t}_i \rightarrow \mathbf{y}_j, \ j = 1, 2, \ldots, N\} \tag{1}$$

Accordingly, the transfer amount $\lambda_{i,j}$ of the jth individual in Φ_i is defined as follows:

$$\lambda_{i,j} = \begin{cases} 1 & if \quad f(\mathbf{y}_j) < f(\mathbf{x}_j) \\ 0 & otherwise \end{cases} \tag{2}$$

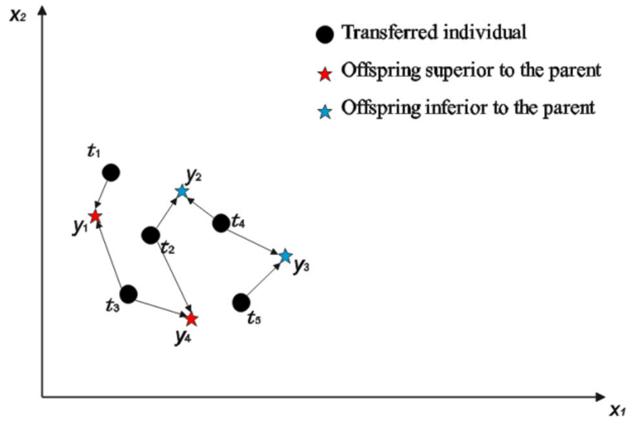


Fig. 2 An instance of the calculation of the transfer ability

where $\mathbf{x}_j$ is the parent of the offspring $\mathbf{y}_j$.

The transfer ability $\omega(t_i)$ of transferred individual t_i is defined as follows, where u represents the size of subset Φ_i.

$$\omega(t_i) = \sum_{j=1}^{u} \lambda_{i,j} \tag{3}$$

Figure 2 shows an example of the calculation of the transfer ability, where all individuals are in a 2-*D* decision space.

In Fig. 2, the historical transferred population TP $= \{\mathbf{t}_1,\mathbf{t}_2,\mathbf{t}_3,\mathbf{t}_4,\mathbf{t}_5\}$ is composed of five transferred individuals. The performance of the offspring $\mathbf{y}_1$ and $\mathbf{y}_4$ is better than that of the parent, while the performance of the offspring $\mathbf{y}_2$ and $\mathbf{y}_3$ is worse than that of the parent. $\mathbf{t}_1$ and $\mathbf{t}_3$ participate in the generation of offspring $\mathbf{y}_1$.$\mathbf{t}_2$ and $\mathbf{t}_4$ participate in the generation of offspring $\mathbf{y}_2$. $\mathbf{t}_4$ and $\mathbf{t}_5$ participate in the generation of offspring $\mathbf{y}_3$. $\mathbf{t}_2$ and $\mathbf{t}_3$ participate in the generation of offspring $\mathbf{y}_4$. Since $\mathbf{t}_1$ only participates in the generation of offspring $\mathbf{y}_1$, and $\mathbf{y}_1$ is superior to the parent, according to Eqs. (1) and (2), the associated subset of $\mathbf{t}_1$ is recorded as $\Phi_1 = \{\mathbf{y}_1\}$; the transfer amount of $\mathbf{t}_1$ is $\lambda_{1,1} = 1$. According to Eq. (3), the transfer ability of $\mathbf{t}_1$ is recorded as $\omega(\mathbf{t}_1) = \lambda_{1,1} = 1$. Similarly, $\mathbf{t}_2$ participates in the generation of the offspring $\mathbf{y}_2$ and $\mathbf{y}_4$. $\mathbf{y}_2$ is inferior to the parent, while $\mathbf{y}_4$ is superior to the parent, then the associated subset of $\mathbf{t}_2$ is recorded as $\Phi_2 = \{\mathbf{y}_2, \mathbf{y}_4\}$, $\lambda_{2,1} = 0$, $\lambda_{2,2} = 1$. According to Eq. (3), the transfer ability of $\mathbf{t}_2$ is $\omega(\mathbf{t}_2) = \lambda_{2,1} + \lambda_{2,2} = 1$. $\mathbf{t}_3$ participates in the generation of the offspring $\mathbf{y}_1$ and $\mathbf{y}_4$. The associated subset of $\mathbf{t}_3$ is recorded as $\Phi_3 = \{\mathbf{y}_1, \mathbf{y}_4\}$. Since the offspring $\mathbf{y}_1$ and $\mathbf{y}_4$ outperform their parents, then $\lambda_{3,1} = 1, \lambda_{3,2} = 1$. The transfer ability of $\mathbf{t}_2$ is $\omega(\mathbf{t}_3) = \lambda_{3,1} + \lambda_{3,2} = 2$. $\omega(\mathbf{t}_4)$ and $\omega(\mathbf{t}_5)$ are calculated in the same way. Table 2 shows the result of transfer ability of the historical transferred population TP.

Table 2 Transfer ability of the population TP

Transferred individual	Associated subset	Transfer amount		Transfer ability
$\mathbf{t}_i$	Φ_i	$\lambda_{i,1}$	$\lambda_{i,2}$	$\omega(\mathbf{t}_i)$
$\mathbf{t}_1$	$\{\mathbf{y}_1\}$	1	/	1
$\mathbf{t}_2$	$\{\mathbf{y}_2, \mathbf{y}_4\}$	0	1	1
$\mathbf{t}_3$	$\{\mathbf{y}_1, \mathbf{y}_4\}$	1	1	2
$\mathbf{t}_4$	$\{\mathbf{y}_2, \mathbf{y}_3\}$	0	0	0
$\mathbf{t}_5$	$\{\mathbf{y}_3\}$	0	/	0

Construction of decision tree

In the proposed EMT-ADT algorithm, the archive A is used to store individuals involved in knowledge transfer. At each generation, n individuals selected from the auxiliary population are used to update the archive. For a multitask problem with two tasks, when optimizing task T_1, the population associated with task T_2 is regarded as an auxiliary population and vice versa. Since it is possible to select individuals with negative transfer by using random selection strategy, a decision tree model classifier is constructed to predict the transfer ability of each individual in the auxiliary population. Subsequently, transferred individuals are sorted in descending order according to transfer ability. The top n transferred individuals with high transfer ability are chosen to be placed into the archive A. Furthermore, the decision tree model classifier is used to predict the transfer ability of individuals in the archive A. The top n transferred individuals with high transfer ability are selected as the historical transferred population *TP* to conduct knowledge transfer for the next generation.

At each generation, the historical transferred population *TP* with *LP* generations is used as the training data, that is TP_DT $= \bigcup_{g=G-LP}^{G-1} TP_g$. The process of the decision tree model classifier construction is described as follows.

1. According to individual transfer ability, the individual with the highest transfer ability in TP_DT is recorded as $\mathbf{t}_{best}$.

$$\mathbf{t}_{best} = \underset{i\in\{1,2,\ldots,m\}}{argmax}\ \omega(\mathbf{t}_i) \tag{4}$$

where m is the size of TP_DT.

2. For $\forall \mathbf{t}_i \in TP\_DT$, it has two feature attributes, Euclidean distance attribute (a_1) and factorial cost attribute (a_2). The Euclidean distance dis($\mathbf{t}_i, \mathbf{t}_{best}$) between $\mathbf{t}_i$ and $\mathbf{t}_{best}$, which is recorded as the data $\mathrm{x}_{i,1}$ of a_1, is put into the set Ω_1. The factorial cost $f(\mathbf{t}_i)$ of individual $\mathbf{t}_i$, which is recorded as the data $\mathrm{x}_{i,2}$ of a_2, is put into the set Ω_2. The transfer ability $\omega(\mathbf{t}_i)$ of individual $\mathbf{t}_i$, which is recorded as the data y_i of label attribute, is put into the set Ω_3. Let $\mathrm{x}_i = \mathrm{x}_{i,1} \cup \mathrm{x}_{i,2}$, the candidate attribute set A $= \{a_1,a_2\}$, the dataset D $= \{(\mathrm{x}_1, \mathrm{y}_1),$

$(x_2, y_2), \ldots, (x_m, y_m)\}$.

$$\mathrm{dis}(\mathbf{t}_i, \mathbf{t}_{best}) = \|\mathbf{t}_i - \mathbf{t}_{best}\|_2 \tag{5}$$

3. Calculate the Gini index of each feature attribute with the sets Ω_1, Ω_2 and Ω_3, where p_k represents the proportion of the current category k.

$$\mathrm{Gini}(a_i) = 1 - \sum_{k=1}^{K} p_k^2 \tag{6}$$

4. Calculate the Gini index of the jth splitting value b_j corresponding to the feature attribute a_i, where v represents the category for the label attribute in dataset D.

$$\mathrm{Gini\_index}(a_i, b_j) = 1 - \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v) \tag{7}$$

5. The splitting value b^* with the lowest Gini index is selected as the optimal splitting attribute. In the candidate attribute set A, the feature attribute a^* corresponding to b^* is selected as the current node.

$$b^* = \mathrm{argmin}\, \mathrm{Gini\_index}(a_i, b_j) \tag{8}$$

Algorithm 2 presents the construction of decision tree.

Algorithm 2. TreeGenerate (D,A)

```
input:   Training set D = {$(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)$}, Attribute set A = {$a_1, a_2$}
output:  A decision tree DT with node as the root node
1:   Generate node
2:   Evaluate the population on task $T_i$, $i = 1,\ldots,m$
3:   if all samples in $D$ belong to the same category $C$
4:       Mark node as a leaf node of category $C$
5:   end if
6:   for $\forall a_i \in A$
7:     Sort the data in $a_i$ in ascending order
8:     The intermediate value $b_j$ of adjacent values is used as the splitting attribute
9:     Calculate the Gini index corresponding to $b_j$ according to Eq.(7)
10:  end for
11:  $b^*$ is selected as the optimal splitting attribute according to Eq.(8), and the
     corresponding $a^*$ is selected as the current node
12:  Split $D$ into two data sets $D_{v1}$ and $D_{v2}$
13:  Call TreeGenerate($D_{v1}$,$A$)
14:  Call TreeGenerate($D_{v2}$,$A$)
```

Take Table 3 as an example to construct a decision tree. Euclidean distance (denoted as *dis*) represents the first feature attribute, while factorial cost (denoted as f) represents the second feature attribute. Firstly, determine the root node. Since the attribute *dis* is a numerical attribute, sort the data in ascending order. Then, the samples are split into two groups with the middle value of adjacent values from small to large. Significantly, two adjacent values are different. For example, if $dis = 0.2$ and $dis = 0.4$, the median value is 0.3. Then, the median value is used as the split point, the calculated Gini index is 0.619. Similarly, other median values and Gini indexes can also be calculated in the same way, as shown in Table 4. In Table 4, the Gini index obtained by taking 0.95 as the split point is the smallest, which is 0.32. Table 5 shows the Gini index of different split points with the factorial cost as node. In Table 5, the Gini index obtained by taking 17.5 as the split point is the smallest, which is 0.441. As seen from Tables 4 and 5, since the Gini index 0.32 in Table 4 is less than the Gini index 0.441 in Table 5, the Euclidean distance is used as the split attribute of the root node, and 0.95 is used as the split value to construct the decision tree. After calculation, the factorial cost is used as the split attribute of intermediate node on the second level, and 10.5 is used as the split value to construct the decision tree. Finally, the constructed decision tree is shown in Fig. 3.

In Fig. 3, the leaf node represents the transfer ability. x_1 represents the first feature attribute: Euclidean distance, while x_2 represents the second feature attribute: factorial cost. Take the transfer ability prediction of data (0.4, 6) as an example, firstly, we judge at the root node. Since 0.4 is less than 0.95, we turn to the left subtree to judge. At the intermediate node, 6 is less than 10.5, the data transfer ability is predicted to be 2.

Table 3 Dataset *D*

dis	0.6	2.4	2.1	0.5	0.4	1.7	1.1	0.8	0.4	0.2	0.8	0.7	1.5	0.6	0.8
f	21	3	7	18	12	16	9	16	1	3	7	9	19	4	2
φ	0	1	1	0	0	1	1	0	2	2	2	2	1	2	2

Table 4 Gini index for feature attribute Euclidean distance

φ	2	0	2	0	0	2	2	0	2	2	1	1	1	1	1
dis	0.2	0.4	0.4	0.5	0.6	0.6	0.7	0.8	0.8	0.8	1.1	1.5	1.7	2.1	2.4
mid	0.3		0.45	0.55		0.65	0.75			0.95	1.3	1.6	1.9	2.25	
Gini	0.619		0.611	0.594		0.541	0.512			0.32	0.412	0.489	0.554	0.61	

Table 5 Gini index for feature attribute factorial cost

φ	2	2	1	2	2	1	2	1	2	0	0	1	0	1	0
f	1	2	3	3	4	7	7	9	9	12	16	16	18	19	21
mid	1.5	2.5		3.5	5.5		8		10.5	14		17.5	18.5	20	
Gini	0.619	0.574		0.585	0.533		0.507		0.444	0.52		0.441	0.621	0.6	

For further explain the prediction of individual transfer ability, a two-task benchmark problem including *Griewank* problem and *Rastrigin* problem is selected. Both problems are characterized by multimodal and nonseparable. The detailed properties are shown as follows.

(1) *Rastrgin*:

$$\mathrm{F(x)} = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10),\ x \in [50,\ 50]^D,\ D = 50 \quad (9)$$

(2) *Griewank*:

$$\mathrm{F}(x) = 1 + \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right),\ x \in [100,\ 100]^D,\ D = 50 \quad (10)$$

Figure 4 shows the decision tree model constructed by the proposed algorithm EMT-ADT at generation *g* on the two-task benchmark problem (Eqs. (9), (10)). At this time, there are six cases of transfer ability (2, 3, 4, 5, 6, 7). Take the transfer ability prediction of data (2.794, 18,694) as an example, firstly, we judge at the root node. Since 2.794 is less than 2.96472, we turn to the left subtree to judge. Similarly, 2.794 is less than 2.95378, we continue to judge on the root node of the left subtree of this node. Since 2.794 is greater than 1.08146, we judge on the root node of the right subtree of this node. Similarly, 2.794 is greater than 2.75214, we continue to judge on the root node of the right subtree of this node. Finally, 18,694 is less than 18,869.9, the transfer ability prediction result of this data is 4.

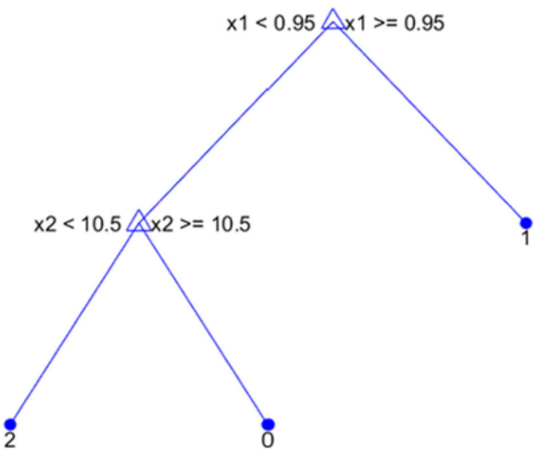


Fig. 3 Decision tree model construction

The transferability of an individual is defined as follows. If an auxiliary parent participates in the generation of an offspring for *m* times, and there are *n* offspring that are superior to the corresponding parent, then the transfer ability of the auxiliary parent is *n*. As seen in Fig. 4, take the leaf node marked in red circle as an example. The node in red circle indicates that an individual reaches the position of the current leaf node through decision tree classification, its corresponding transferability is 7.

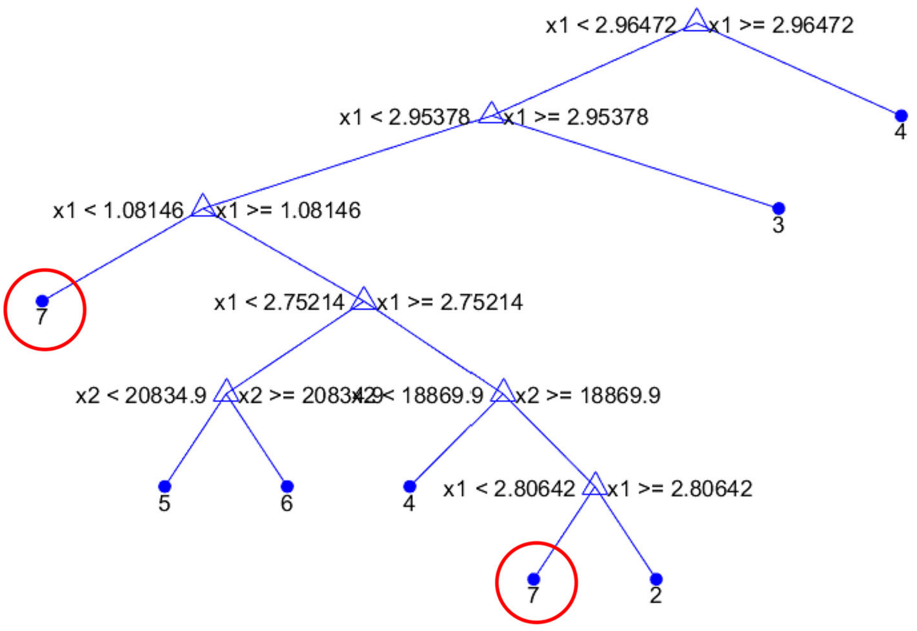


Fig. 4 Decision tree model construction for a two-task benchmark problem

Search strategy

Many classical optimization algorithms, such as GA (genetic algorithm), DE (differential evolution), PSO (particle swarm optimization), TLBO (teaching–learning-based optimization), and BSO (brain storm algorithm) can be used as search engine in MFO paradigm [22–25]. Different algorithms have different search performance. Obviously, well designed search strategies can improve the search efficiency. The success-history based adaptive differential evolution algorithm (SHADE) proposed by Tanabe et al. [26] has been proved to be an effective optimization algorithm. In SHADE, a historical memory is used to store the control parameters that performed well during the evolution. New control parameters are generated by sampling the parameters in the historical memory, which may further improve the performance of the algorithm. Since SHADE outperformed many state-of-the-art DE algorithms on CEC2013 benchmark set and CEC2005 benchmarks, this paper selects SHADE as the search engine. The mutation operator is defined as follows [26].

$$\mathbf{v}_i = \mathbf{x}_i + F_i\left(\mathbf{x}_{pbest} - \mathbf{x}_i\right) + F_i(\mathbf{x}_{r1} - \widetilde{\mathbf{x}}_{r2}) \tag{11}$$

where $\mathbf{x}_i$ is the ith individual of the current population. $\mathbf{x}_{pbest}$ is randomly selected from the top $p\%$ individuals in the current population, while $\mathbf{x}_{r1}$ is randomly selected from the current population.$\widetilde{\mathbf{x}}_{r2}$ is randomly selected from the union of the current population and the archive. The details of SHADE can be found in [26].

To improve the knowledge transfer between different tasks on MFO problems, the original mutation operator of SHADE is modified and is defined as follows.

$$\mathbf{v}_i = \mathbf{x}_i + F_i\left(\mathbf{x}_{pbest} - \mathbf{x}_i\right) + F_i(\mathbf{x}'_{r1} - \mathbf{x}'_{r2}) \tag{12}$$

For a multitask problem with two tasks, P is the subpopulation corresponding to the target task, and P' is the subpopulation corresponding to the auxiliary task. TP is the historical transferred population, which is used to provide the transferred individuals for the target task. $\mathbf{x}_i$ is the ith individual of the population P. $\mathbf{x}_{pbest}$ is randomly selected from the top $p\%$ individuals in the population P'. $\mathbf{x}'_{r1}$ and $\mathbf{x}'_{r2}$ are randomly selected from TP. F is the scale factor. After mutation operator, the same crossover operator as in SHADE is used to generate the final offspring.

In the modified SHADE mutation operator, $\mathbf{x}_{pbest}$ can provide the promising direction for the population, which can promote the convergence. The randomly constructed difference vector $(\mathbf{x}'_{r1} - \mathbf{x}'_{r2})$ not only enhances the population diversity, but also promote positive knowledge transfer due to the selection of the transferred individuals with highest transfer ability.

The proposed EMT-ADT

The pseudocode of the EMT-ADT algorithm is shown in Algorithm 3. P' is the subpopulation corresponding to the auxiliary task. The archive A is used to store individuals involved in knowledge transfer. $\mathbf{x}_{best}$ is the best individual of P'. NP_i is the population size of ith task. FES and $MAXFES$ represent the current number of function evaluations and the maximal number of function evaluations, respectively.

The main steps of EMT-ADT is as follows. First, for each task T_i, the offspring is generated by Eq. (11) or (12) according to the random mating probability *rmp*. Meanwhile, the associated subset Φ_i, the transfer amount $\lambda_{i,j}$ and the transfer ability $\omega(t_i)$ are calculated according to Eqs. (1)–(3). When knowledge transfer is required during the generation of offspring, the training data *D* is first constructed by *TP* of continuous *LP* generation. Next, the decision tree model *DT* is constructed by the training data. Then, *DT* is used to predict the transfer ability of all individuals in P'. All individuals are sorted based on the transfer ability. The top *n*-1 individuals and the historical best individual in P' are stored into the archive *A*. Then, *DT* is used to predict the transfer ability of all individuals in the archive *A*. After ranking these individuals according to their transfer ability, the top $n-1$ individuals and the historical best individual in P' are stored into the historical transferred population TP_{g+1} at generation $g+1$. When there is no knowledge transfer during the generation of offspring, the top $n-1$ individuals with better factorial cost and the historical best individual in P' are stored into the archive *A*. The individuals in the archive *A* are sorted according to their transfer ability. Then, the top $n-1$ individuals and the historical best individual in *A* are selected as the historical transferred population TP_{g+1} at generation $g+1$.

When the success rate sr_i (sr_i denotes the rate that offspring is better than its parent) is greater than the given threshold, the random mating probability rmp_i for each task is updated as follows [10].

$$rmp_i = \begin{cases} min\{rmp_i + 0.3(1-sr_i),\ 1]\} & if\ TNP = 0 \\ \left\lfloor \frac{tsr_i}{tsr_i+sr_i} + 0.5 \right\rfloor \times \min\{rmp_i + 0.3 \times tsr_i,\ 1\} & \\ \quad + \left\lfloor \frac{sr_i}{tsr_i+sr_i} + 0.5 \right\rfloor \times max\{rmp_i - 0.3 \times (1-tsr_i),\ 0\} & otherwise \end{cases}, \tag{13}$$

where tsr_i denotes the rate that offspring generated with knowledge transfer is better than its parent. If all offspring are generated without knowledge transfer, then *TNP* is set to 0; otherwise, *TNP* is the number of offspring generated with knowledge transfer.

Algorithm 3 EMT-ADT algorithm

```
1:  Initialize population $\mathbf{P}_i$ for each task $T_i$ and evaluate the individuals in $\mathbf{P}_i$ ($i$=1,2,..,$m$)
2:  Initialize historical transferred population TP and $rmp_i$ ($i$=1,2,..,$m$)
3:  while the termination criteria are not met
4:      for each task $T_i$
5:          Set $flag_j$ = 0 (j=1,2,..,$NP_i$), $TNP$=0
6:          for each $\mathbf{x}_j \in \mathbf{P}_i$
7:              if $rand < rmp_i$
8:                  Generate offspring according to Eq.(12)
9:                  $flag_j$ = 1, $TNP$= $TNP$+1
10:                 Construct the associated subset $\Phi_i$
11:             else
12:                 Generate offspring according to Eq.(11)
13:             end if
14:         end for
15:         Calculate $sr_i$ and $tsr_i$
16:         Update $P_i$, $rmp_i$ and transfer ability $\varphi(\mathbf{t})$ of each transferred individual in $TP$
17:         Update the training data TP_DT
18:         Construct the dataset $\boldsymbol{D}$
19:         if $\sum_{j=1}^{NP_i} flag_j > 0$
20:             Construct the decision tree $DT$ according to Algorithm 2
21:             Predict the transfer ability of all individuals in $P'$ with $DT$
22:             Sort the individuals according to the transfer ability
23:             Select the top $n$-1 individuals → A
24:             $\mathbf{A} = \mathbf{A} \cup \mathbf{x}_{best}$
25:             Predict the transfer ability of all individuals in A with $DT$
26:             Sort the individuals in A according to the transfer ability
27:             Select the top $n$-1 individuals → $\mathbf{TP}_{g+1}$
28:             $\mathbf{TP}_{g+1} = \mathbf{TP}_{g+1} \cup \mathbf{x}_{best}$
29:         else
30:             Select $n$-1 individuals in $P'$ with better factorial cost → A
31:             $\mathbf{A} = \mathbf{A} \cup \mathbf{x}_{best}$
32:             Sort the individuals in A according to the transfer ability
33:             Select the top $n$-1 individuals → $\mathbf{TP}_{g+1}$
34:             $\mathbf{TP}_{g+1} = \mathbf{TP}_{g+1} \cup \mathbf{x}_{best}$
35:         end if
36:     end for
37:     if $FES/MAXFES$ == cos($\gamma$)/4
38:         $N = N/2$
39:     end if
40: end while
41: Output the best individual for each task
```

Complexity analysis

The computational cost of the EMT-ADT mainly comes from assortative mating, the adaptive knowledge transfer based on decision tree and historical transferred population update. In EMT-ADT, a loop over *NP* (population size) is conducted, containing a loop over D (dimension) and m optimization tasks. Assortative mating is performed according to the random mating probability (*rmp*). Then, the runtime complexity is $O(m \cdot NP \cdot D)$ at each iteration. For the adaptive knowledge transfer based on decision tree, the Euclidean distance between the transferred individual r_i and the optimal transferred individual r_{best} is calculated, which may increase the time complexity of the algorithm. Due to five consecutive iterations, the runtime complexity is $O(5 \cdot n \cdot D)$, in which n is the number of the transferred individuals. Similarly, the runtime complex of historical transferred population update is $O(n \cdot D)$. Therefore, the total time complexity of the EMT-ADT is $O(m \cdot NP \cdot D)$.

Table 6 Properties of problem pairs for CEC2017 multitask problems

Intersection degree	Problem	Task group		Similarity
		T_1	T_2	
Complete	CI + HS	Griewank	Rastrigin	High (1.0000)
	CI + MS	Ackley	Rastrigin	Medium (0.2261)
	CI + LS	Ackley	Schwefel	Low (0.0002)
Partial	PI + HS	Rastrigin	Sphere	High (0.8670)
	PI + MS	Ackley	Rosenbrock	Medium (0.2154)
	PI + LS	Ackley	Weierstrass	Low (0.0725)
No	NI + HS	Rosenbrock	Rastrigin	High (0.9434)
	NI + MS	Griewank	Weierstrass	Medium (0.3669)
	NI + LS	Rastrigin	Schwefel	Low (0.0016)

Comparative studies of experiments

To evaluate the competitiveness of the proposed EMT-ADT algorithm, 9 benchmark test problems from the CEC2017 Evolutionary Multi-Task Optimization Competition [27] are employed. Each test problem is a two-task problem. According to the similarity between the landscapes of two tasks, the benchmark problems can be categorized into three groups: high similarity (*HS*), medium similarity (*MS*) and low similarity (*LS*). Furthermore, the benchmark problems can be divided into three groups according to the intersection degree of the global optima: complete intersection (*CI*), partial intersection (*PI*) and no intersection (*NI*). The details of these benchmark problems can be found in [27]. In addition, two complex single-objective MFO test suites, i.e. WCCI20-MTSO and WCCI20-MaTSO, are selected to further verity the competitiveness of the proposed EMT-ADT. Both WCCI20-MTSO and WCCI20-MaTSO include 10 test problems, which are put forward for WCCI 2020 Competition on Evolutionary Multitasking Optimization [28]. Each test problem in WCCI20-MTSO is with 2 tasks, while each test problem in WCCI20-MaTSO is with 10 tasks.

Parameter settings

The aforementioned CEC2017 multitask problems are given in Table 6.

The proposed EMT-ADT is compared with eight popular multitask optimization algorithms, namely MFEA [1], MFEARR (MFEA with resource reallocation) [29], MFDE (multifactorial differential evolution) [23], AT-MFEA (affine transformation-enhanced MFEA) [8], SREMTO [13], MFMP (MFO via explicit multipopulation evolutionary framework) [10], TLTLA (two-level transfer learning algorithm) [30] and MTEA-AD (MTEA based on anomaly detection) [31]. The default parameter settings for these algorithms are given in Table 7. *N* denotes the population size.

Table 7 Default parameters settings of compared algorithms

Algorithm	*N*	*rmp*	Parameter
MFEA	100	0.3	$mu = 2;\ mum = 5$
MFEARR	100	0.3	$mu = 2;\ mum = 5;\ \varepsilon = 0.01$
MFDE	100	0.3	$F = 0.5;\ CR = 0.9;$
AT-MFEA	100	0.3	$pc = 1;\ \eta c = 15;\ pm = 0.02;\ \eta m = 15;\ \alpha = 0.5$
SREMTO	100	0.3	$P_\alpha = 0.7;\ P_\beta = 1.0$
MFMP	200	/	$\theta = 0.2;\ c = 0.3;\ \alpha = 0.25$
TLTLA	100	0.3	$mu = 2;\ mum = 5$
MTEA-AD	100	0.1	$pc = 1;\ \eta c = 2;\ pm = 0.02;\ \eta_m = 5;\ \alpha = 0.25$
EMT-ADT	200	/	$n = 10;\ \gamma = 0.001$

Experiments on CEC2017 multitask problems

In this section, the proposed EMT-ADT algorithm is compared with the above-listed algorithms to verify the performance. Tables 8, 9, 10 summarize the mean fitness (mean) and standard deviation (Std) achieved by MFEA, MFEARR, MFDE, AT-MFEA, SREMTO, MFMP, TLTLA, MTEA-AD and EMT-ADT over 30 runs. The codes are conducted with *MAXFES* as the termination criterion, which is set to 200,000. Three statistical test measures including the Wilcoxon signed-rank test [32], the multiple-problem Wilcoxon's test [33] and Friedman's test [33] are used to compare EMT-ADT with other eight algorithms. With regard to the single-problem Wilcoxon's test, "†", "≈" and "−" are used to indicate that EMT-ADT significantly wins, equal, and is worse than the compared algorithm, respectively. With regard to the multiple-problem Wilcoxon's test, R^+ and R^- are used to indicate that EMT-ADT is significantly better than or worse than the compared algorithm, respectively. The best solution is highlighted in bold.

Table 8 Mean fitness values and Std obtained by MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO, MFMP and EMT-ADT on complete intersection problems

Algorithm	Index	CI-HS		CI-MS		CI-LS		Summary
		T_1	T_2	T_1	T_2	T_1	T_2	†/≈/–
MFEA	Mean	8.84E−02†	1.63E + 02†	4.79E + 00†	1.95E + 02†	2.01E + 01†	2.97E + 03†	6/0/0
	Std	1.90E−02	5.59E + 01	8.96E−01	5.27E + 01	4.58E−02	4.05E + 02	
MFEARR	Mean	6.57E−02†	4.40E + 02†	1.46E + 01†	4.58E + 02†	2.01E + 01†	2.64E + 03†	6/0/0
	Std	1.39E−02	8.63E + 01	7.82E + 00	7.20E + 01	8.66E−02	4.28E + 02	
AT-MFEA	Mean	7.10E−03†	3.65E + 01†	2.58E + 00†	1.03E + 02†	2.06E + 01†	2.68E + 03†	6/0/0
	Std	9.60E−03	5.22E + 01	5.22E−01	3.31E + 01	5.40E−01	4.09E + 02	
MTEA-AD	Mean	1.38E−02†	9.27E + 01†	3.10E + 00†	1.66E + 02†	2.04E + 01†	2.52E + 03†	6/0/0
	Std	1.16E−02	1.03E + 02	7.07E−01	5.27E + 01	5.20E−01	4.94E + 02	
MFDE	Mean	7.55E−16†	7.35E−13†	3.06E−08†	7.33E−12†	2.11E + 01†	7.13E + 03†	6/0/0
	Std	2.81E−15	3.42E−12	1.13E−07	2.89E−11	2.20E−01	1.73E + 03	
TLTLA	Mean	5.84E−06†	9.30E−03†	4.50E−03†	1.59E−02†	1.48E + 01†	9.94E + 02†	6/0/0
	Std	1.18E−05	1.84E−02	2.60E−03	2.01E−02	9.21E + 00	5.19E + 02	
SREMTO	Mean	8.30E−03†	5.39E + 01†	1.09E + 01†	3.97E + 02†	1.58E + 01†	1.15E + 04†	6/0/0
	Std	1.05E−02	7.52E + 01	6.48E + 00	1.98E + 02	3.03E + 00	2.18E + 03	
MFMP	Mean	0.00E + 00≈	4.74E−15≈	2.66E−15†	0.00E + 00≈	1.82E + 01†	2.37E + 01†	3/3/0
	Std	0.00E + 00	1.84E−14	3.46E−15	0.00E + 00	6.77E + 00	6.52E + 01	
EMT-ADT	Mean	0.00E + 00	0.00E + 00	8.88E−16	0.00E + 00	3.36E−03	6.36E−04	/
	Std	0.00E + 00	0.00E + 00	1.04E−31	0.00E + 00	1.50E−03	0.00E + 00	

Table 9 Mean fitness values and Std obtained by MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO, MFMP and EMT-ADT on partial intersection problems

Algorithm	Index	PI-HS		PI-MS		PI-LS		Summary
		T_1	T_2	T_1	T_2	T_1	T_2	†/≈/–
MFEA	Mean	5.43E + 02†	3.55E−01†	3.13E + 00†	2.27E + 02†	1.94E + 01†	1.92E + 01†	6/0/0
	Std	1.15E + 02	9.85E−02	7.31E−01	5.92E + 01	2.96E + 00	4.06E + 00	
MFEARR	Mean	4.39E + 02†	1.40E−01†	1.68E + 01†	7.67E + 02†	1.29E + 01†	1.80E + 01†	6/0/0
	Std	5.88E + 01	3.61E−02	6.43E + 00	1.23E + 03	8.20E + 00	2.59E + 00	
AT-MFEA	Mean	2.47E + 02†	2.00E−03†	2.57E + 00†	1.37E + 02†	2.75E + 00†	3.09E + 00†	6/0/0
	Std	5.00E + 01	1.20E−03	5.03E−01	3.42E + 01	4.64E−01	9.89E−01	
MTEA-AD	Mean	3.29E + 02†	2.27E−02†	2.84E + 00†	1.41E + 02†	3.11E + 00†	3.86E + 00†	6/0/0
	Std	6.18E + 01	1.61E−02	4.71E−01	3.35E + 01	4.33E−01	9.52E−01	
MFDE	Mean	7.67E + 01≈	2.66E−15†	9.52E−07†	6.84E + 01†	1.47E−03†	4.76E−04†	5/1/0
	Std	1.98E + 01	1.34E−14	3.64E−06	1.99E + 01	4.70E−03	1.40E−03	
TLTLA	Mean	1.77E + 01 –	4.17E−02†	1.99E + 00†	4.80E + 01†	9.58E−02†	2.40E−03†	5/0/1
	Std	5.49E + 01	1.62E−02	3.78E−01	2.20E + 01	4.08E−02	6.20E−03	
SREMTO	Mean	4.25E + 02†	3.11E−05†	7.88E + 00†	2.48E + 02†	1.43E + 01†	1.00E + 01†	6/0/0
	Std	1.48E + 02	6.78E−05	5.10E + 00	1.92E + 02	7.19E + 00	4.49E + 00	
MFMP	Mean	1.16E + 02†	7.47E−28†	1.27E−14†	1.92E + 01†	1.84E−15†	**1.99E−18**≈	5/1/0
	Std	1.30E + 01	6.94E−28	4.41E−15	1.59E + 01	3.09E−15	0.00E + 00	
EMT-ADT	Mean	7.72E + 01	0.00E + 00	7.99E−15	4.63E + 00	8.88E−16	1.99E−18	/
	Std	1.28E + 01	7.23E−28	5.75E−15	4.40E + 00	1.42E−31	0.00E + 00	

Table 10 Mean fitness values and Std obtained by MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO, MFMP and EMT-ADT on no intersection problems

Algorithm	Index	NI-HS		NI-MS		NI-LS		Summary
		T_1	T_2	T_1	T_2	T_1	T_2	†/≈/–
MFEA	Mean	2.48E + 02†	2.18E + 02†	1.03E−01†	2.68E + 01†	5.78E + 02†	2.85E + 03†	6/0/0
	Std	6.71E + 01	5.40E + 01	2.24E−02	2.47E + 00	1.22E + 02	4.17E + 02	
MFEARR	Mean	5.89E + 02†	4.30E + 02†	6.78E−02†	4.59E + 01†	4.38E + 02†	2.84E + 03†	6/0/0
	Std	1.22E + 03	7.46E + 01	1.70E−02	3.82E + 00	9.83E + 01	4.88E + 02	
AT-MFEA	Mean	1.42E + 02†	1.76E + 02†	1.01E−02†	2.22E + 01†	2.52E + 02†	2.54E + 03†	6/0/0
	Std	3.71E + 01	5.72E + 01	6.90E−03	4.04E + 00	5.17E + 01	4.37E + 02	
MTEA-AD	Mean	1.60E + 02†	2.45E + 02†	1.90E−02†	2.38E + 01†	3.03E + 02†	2.76E + 03†	6/0/0
	Std	4.98E + 01	5.10E + 01	9.40E−03	4.16E + 00	5.01E + 01	3.43E + 02	
MFDE	Mean	6.80E + 01†	2.42E + 01†	9.86E−04†	3.21E + 00†	9.11E + 01†	4.01E + 03†	6/0/0
	Std	3.06E + 01	1.37E + 01	3.20E−03	9.61E−01	2.34E + 01	7.60E + 02	
TLTLA	Mean	4.53E + 01†	1.28E−01†	1.72E−04†	2.43E−01 –	3.69E−01 –	7.87E + 02†	4/0/2
	Std	6.70E + 00	4.48E−01	2.20E−04	1.17E−01	5.77E−01	4.92E + 02	
SREMTO	Mean	6.42E + 02†	3.26E + 02†	1.42E−01†	2.75E + 01†	5.31E + 02†	1.19E + 04†	6/0/0
	Std	1.07E + 03	2.47E + 02	7.33E−01	4.20E + 00	2.14E + 02	5.55E−12	
MFMP	Mean	1.71E + 01†	1.33E−14†	1.34E−15†	1.84E + 00†	1.09E + 02†	4.34E + 01†	6/0/0
	Std	1.13E + 01	2.77E−14	8.91E−16	6.92E−01	1.85E + 01	6.59E + 01	
EMT-ADT	Mean	3.88E + 00	0.00E + 00	5.26E−16	3.10E−01	4.87E + 01	6.36E−04	/
	Std	3.38E + 00	0.00E + 00	2.89E−16	4.89E−01	1.05E + 01	2.93E−12	

Comparisons on complete intersection problems

Table 8 shows that the proposed EMT-ADT algorithm outperforms MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA and SREMTO on complete intersection problems, which indicates that the decision tree model prediction strategy employed in the proposed algorithm work effectively and efficiently. More specially, EMT-ADT and MFMP have similar performance on CI + HS and CI + MS problems, while EMT-ADT performs better than MFMP on CI + LS problems. Furthermore, the experimental results show that the proposed EMT-ADT algorithm wins the MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO and MFMP on 6, 6, 6, 6, 6, 6, 6, 3 optimization tasks, respectively. The convergence curves of the mean fitness values obtained by each algorithm on complete intersection problems are plotted in Fig. 5. As can be seen in Fig. 5, EMT-ADT algorithm converges faster than the competitive algorithms.

Comparisons on partial intersection problems

As can be observed in Table 9, the proposed EMT-ADT algorithm outperforms the MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO and MFMP in 4 out of 6 tasks. TLTLA performs best on task T_1 of PI + HS problem. For the task T_2 of PI + LS problem, the performance of EMT-ADT is similar to that of MFMP. All in all, the results show that the proposed EMT-ADT outperforms the MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO and MFMP on 6, 6, 6, 6, 5, 5, 6, 5 optimization tasks, respectively. Figure 6 shows the convergence curves of the mean fitness values obtained by each algorithm on partial intersection problems. As can be seen in Fig. 6, compared with the eight competitive algorithms, EMT-ADT algorithm converges faster on all tasks except task T_1 of PI + HS problem.

Comparisons on no intersection problems

Table 10 shows that the proposed EMT-ADT algorithm outperforms the MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO and MFMP in 4 out of 6 tasks. For task T_2 of NI-MS problem and task T_1 of NI-LS, TLTLA ranked first while EMT-ADT ranked second. The results show that the proposed EMT-ADT outperforms the MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO and MFMP on 6, 6, 6, 6, 6, 4, 6, 6 optimization tasks, respectively. Figure 7 shows the convergence curves of the mean fitness values obtained by each algorithm on no intersection problems. As can be seen in Fig. 7, compared with the eight competitive algorithms, EMT-ADT algorithm

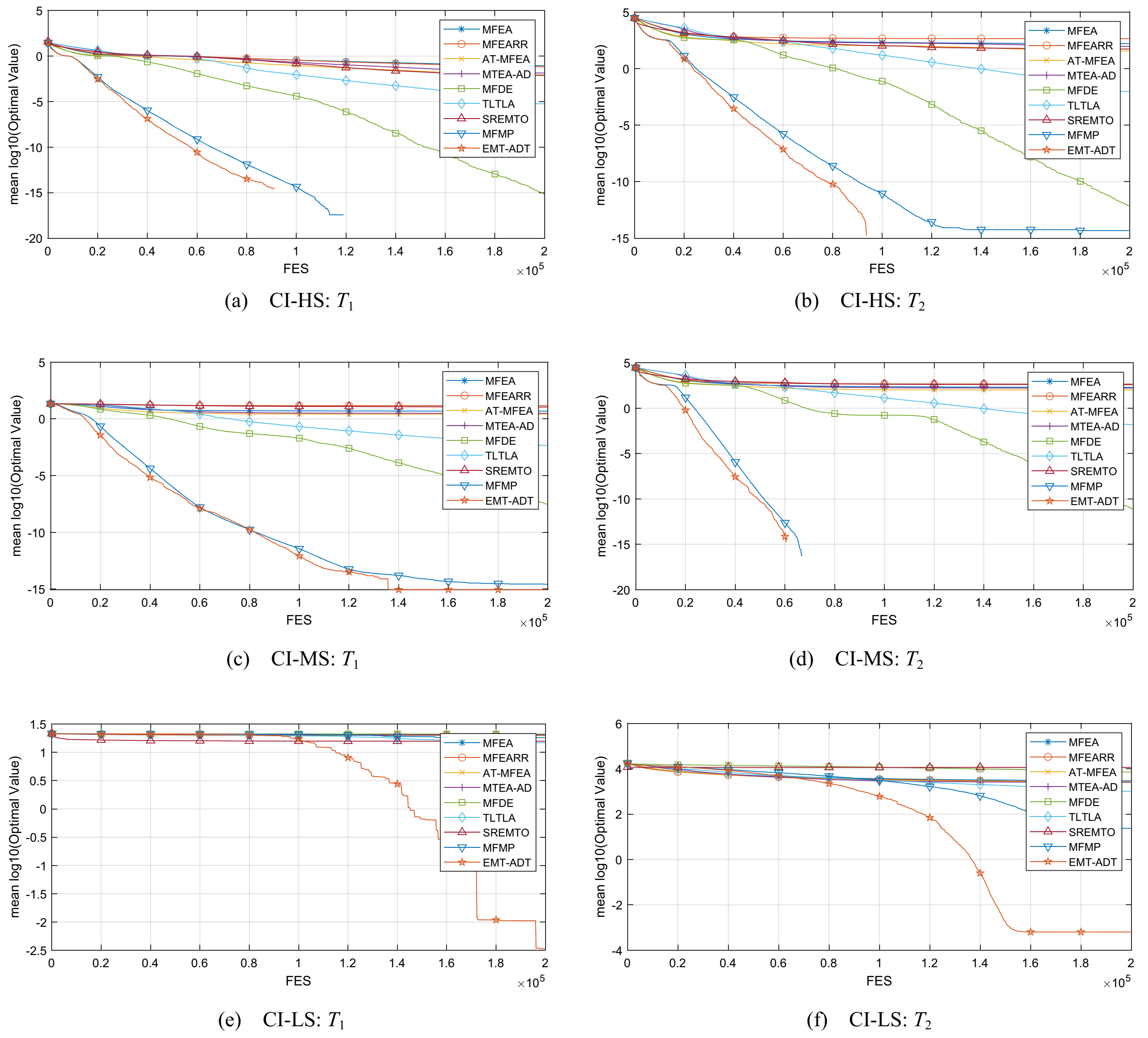


Fig. 5 Convergence curves of the average fitness values obtained by MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO, MFMP and EMT-ADT on complete intersection problems

converges faster on all tasks except task T_2 of NI + MS problem and task T_1 of NI + LS problem.

Adaptive knowledge transfer analysis

To evaluate the effectiveness of the proposed adaptive knowledge transfer, EMT-ADT is compared with the algorithm without adaptive knowledge transfer (EMT) on 3 representative CEC2017 multitask problems including CI + MS, PI + MS and NI + MS. As explained before, if the offspring generated with adaptive knowledge transfer has better performance than its parent, it is referred to as positive transfer. Figure 8 shows the convergence curves of the mean fitness values obtained by EMT-ADT and EMT on test problems CI + MS, PI + MS and NI + MS. Figure 9 shows the mean number of the transferred individuals during evolution in EMT-ADT on test problems CI + MS, PI + MS and NI + MS. In Fig. 9, red circles and blue stars denote the number of transferred individuals in the process of evolution on task T_1 and task T_2, respectively. As seen in Figs. 8 and 9, EMT-ADT outperforms EMT in terms of the solution quality, which demonstrates that adaptive knowledge transfer strategy can improve the performance of the algorithm.

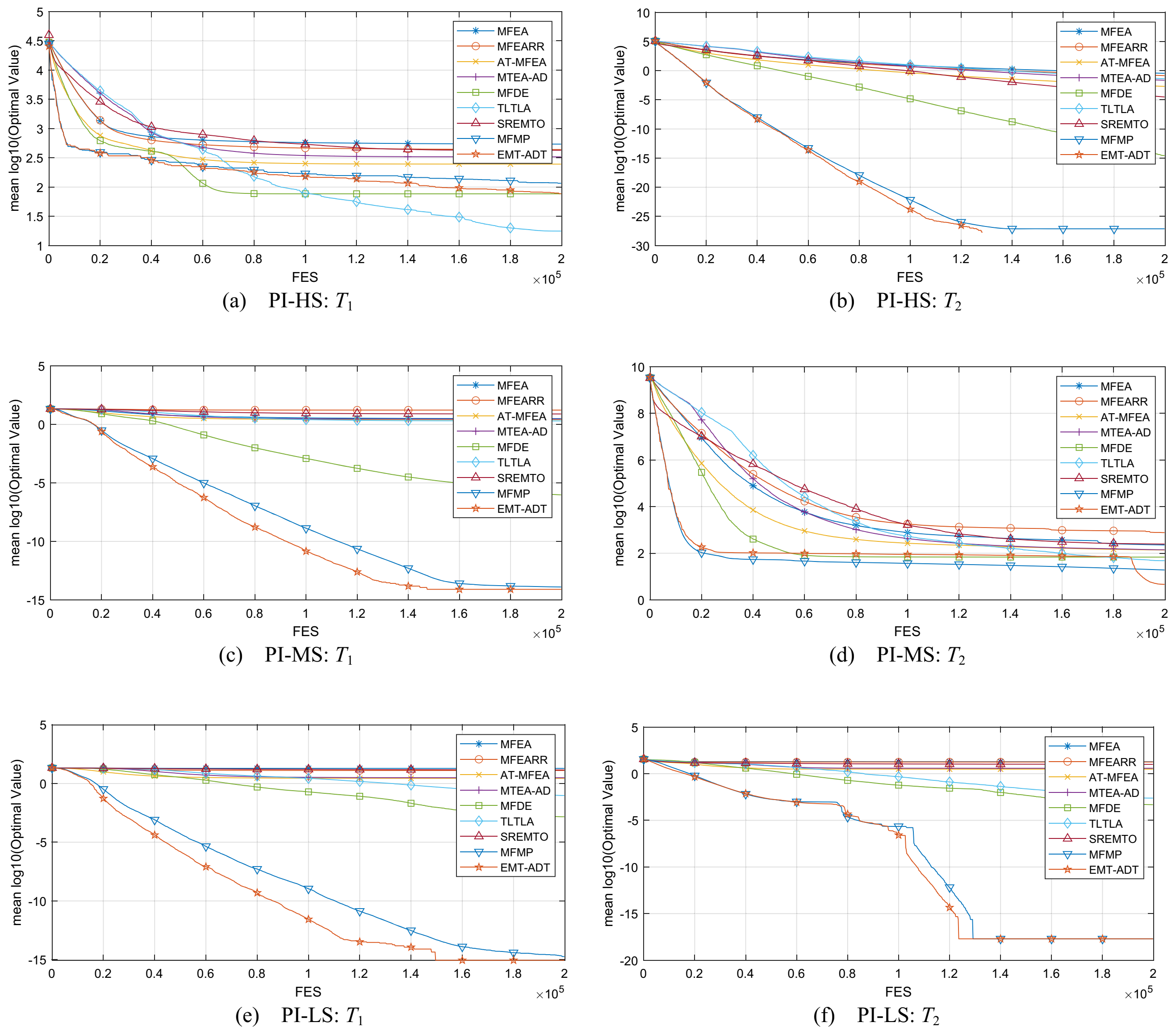


Fig. 6 Convergence curves of the average fitness values obtained by MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO, MFMP and EMT-ADT on partial intersection problems

Comparisons on WCCI20-MSTO

To investigate the performance on more complex multitasking problems, the proposed EMT-ADT is further compared with eight state-of-the-art multitasking optimization algorithms on WCCI20-MSTO benchmark suite. Tables 11 and 12 show the comparative results of nine algorithms in terms of mean fitness.

As can be seen from Tables 11 and 12, EMT-ADT obtained 17 best values out of 20 WCCI20-MTSO test instances. MFEA, MFEARR and MTEA-AD perform well on problem P_8. SREMTO achieves the best value on the Task T_1 of problem P_9. The last rows of Tables 11 and 12 imply that EMT-ADT performs significantly better than the eight competitors over almost all the test instances.

Figure 10 plots the trajectory of the mean fitness versus the number of function evaluations in each algorithm on WCCI20-MSTO benchmark suite. As seen in Fig. 10, EMT-ADT demonstrates a clear advantage over eight competitors.

Comparisons on WCCI20-MaTSO

The mean fitness values obtained by nine algorithms on WCCI20-MaTSO are shown in Table 13. It is clear that EMT-ADT show obvious advantage over the competitors. More specifically, EMT-ADT significantly outperforms the eight competitors on 94 instances out of 100 instances. The

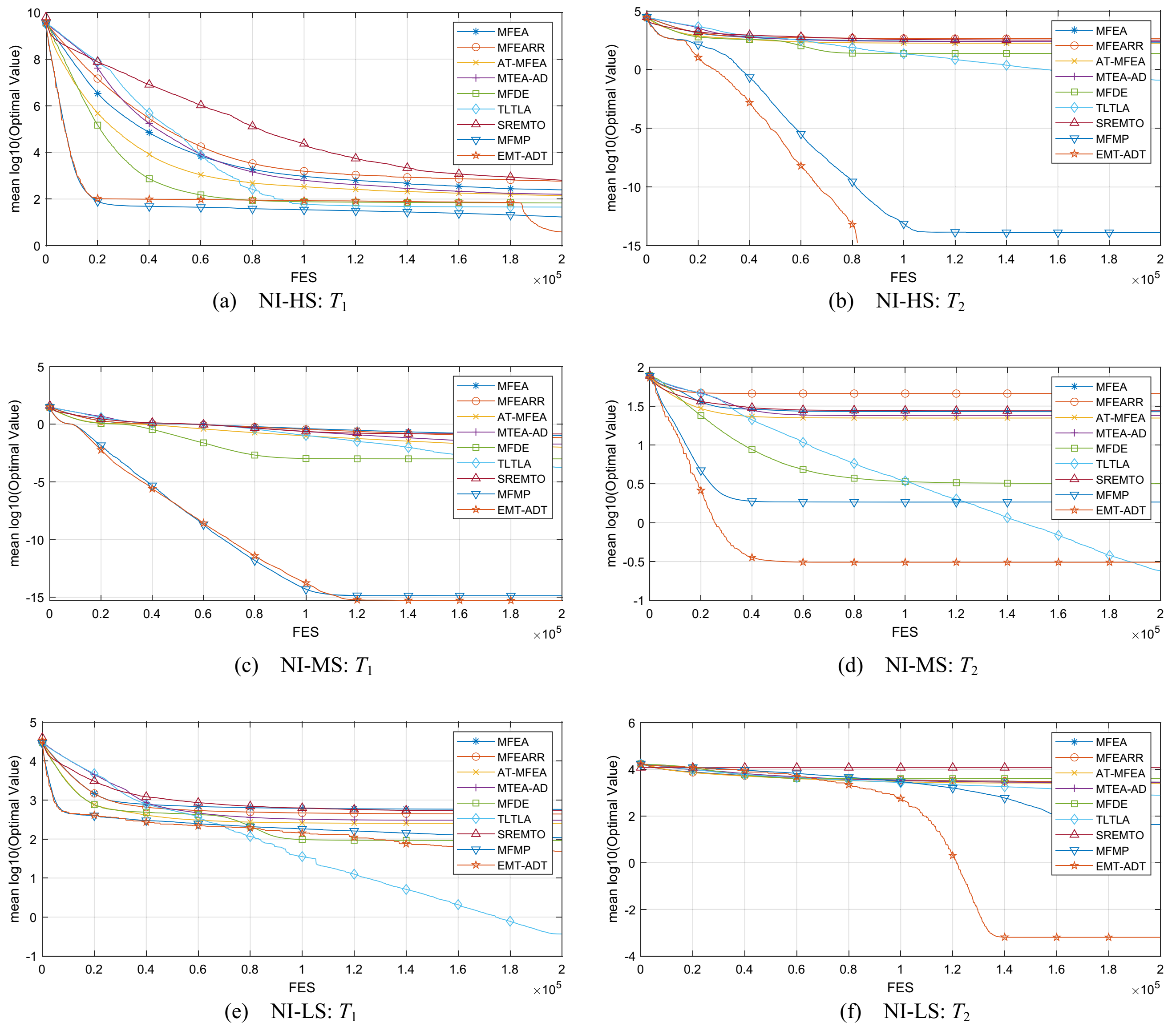


Fig. 7 Convergence curves of the average fitness values obtained by MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO, MFMP and EMT-ADT on no intersection problems

last row of Table 13 shows that the proposed EMT-ADT outperforms the MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO and MFMP on 94, 94, 95, 94, 100, 94, 94 and 100 optimization tasks, respectively.

Comparisons with single-task algorithms

Multitasking optimization algorithm makes full use of positive knowledge based on sharing solutions across tasks, which can improve convergence and solution quality. To verify the competitiveness of the proposed algorithm, a series of experiments are conducted against two state-of-the-art single-task methods, i.e. SHADE and LSHADE. SHADE is an adaptive DE characterized by introducing success-history based parameter adaptation, while LSHADE is an improvement of SHADE. The performance of EMT-ADT, SHADE and LSHADE on CEC2017 benchmark problems and WCCI20_MTSO benchmark problems are summarized in Tables 14 and 15.

As can be observed in Table 14, EMT-ADT achieves superior performance in terms of solution quality on almost all the CEC2017 benchmarks. More specifically, the proposed EMT-ADT wins 17 out of the 18 competitions, which demonstrates that the implicit knowledge transfer across tasks in multitasking is beneficial for improving the performance of EMT-ADT. From Table 15, it can be seen that EMT-ADT works well on more complex multitasking benchmark suits

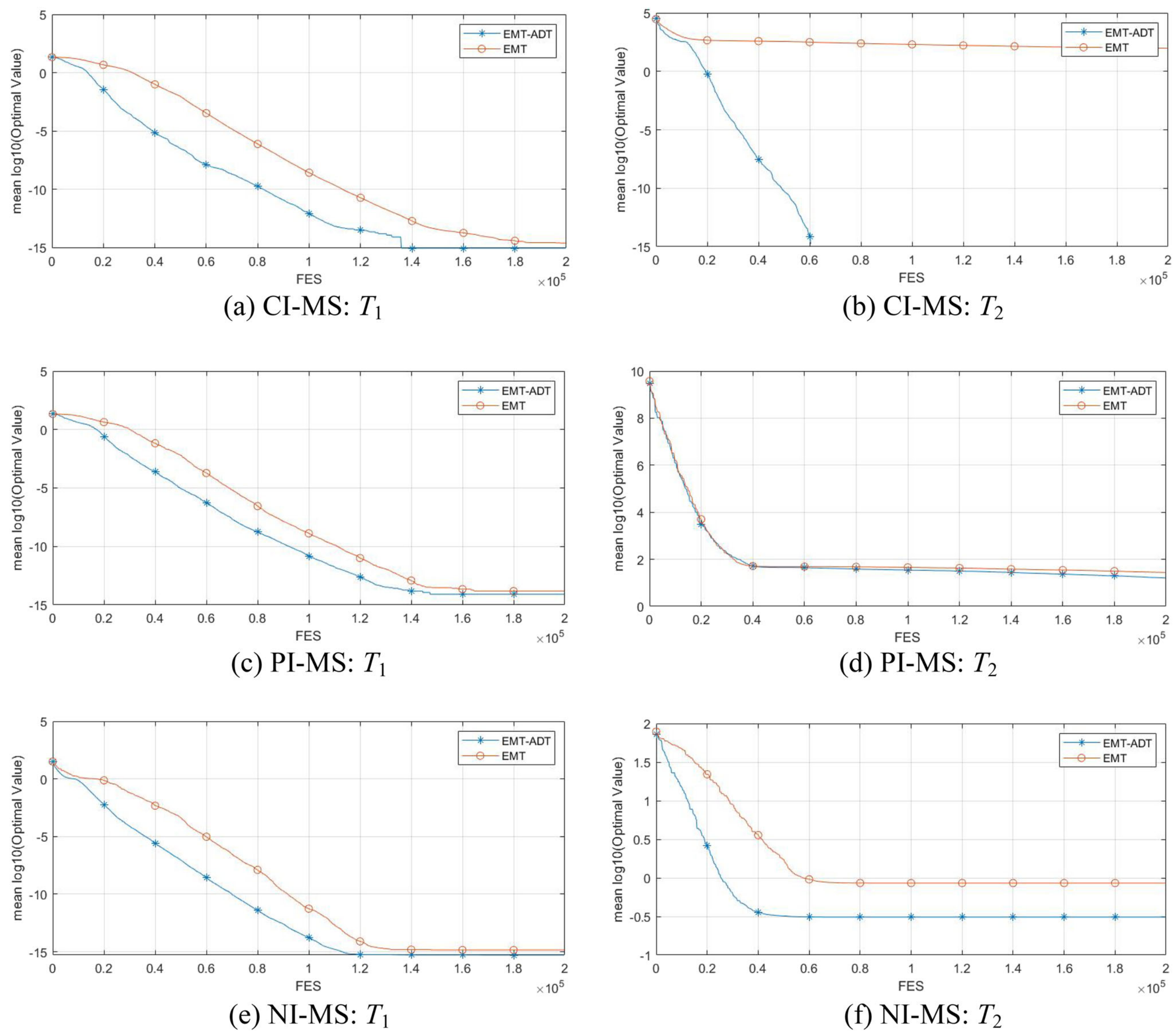


Fig. 8 Convergence curves of the mean fitness values obtained by EMT-ADT and EMT on CI + MS, PI + MS and NI + MS

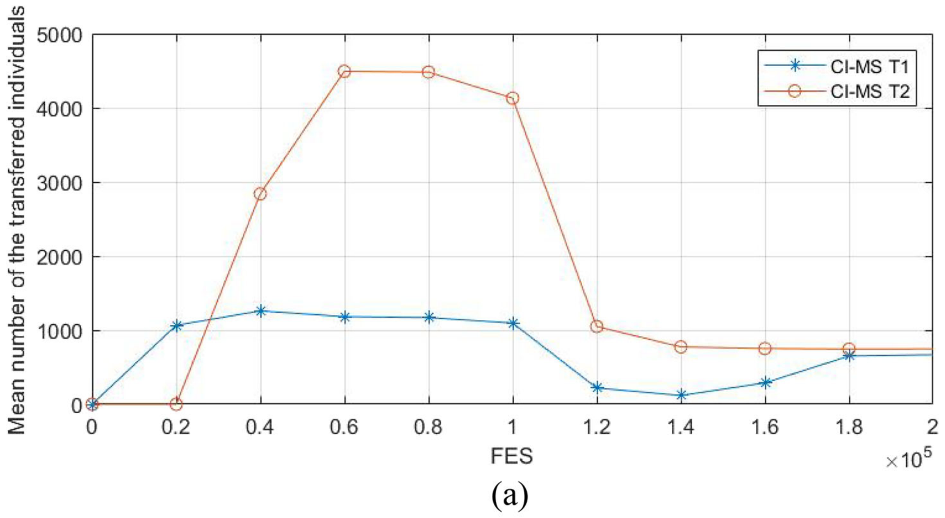
WCCI20_MTSO, better than the values obtained by SHADE and LSHADE, i.e., SHADE and LSHADE got the best result on 0 and 3 instances.

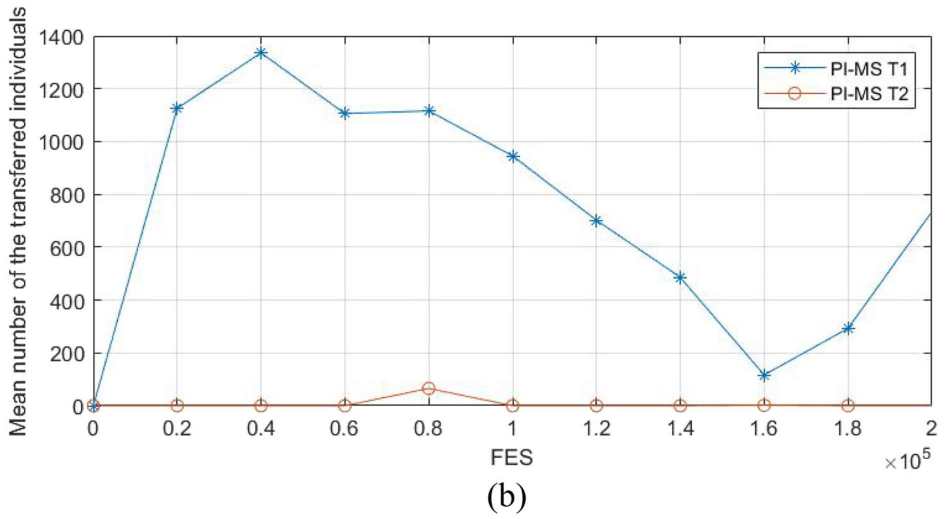
Statistical analysis

In this section, the multiple-problem Wilcoxon's test is used to compare the significant difference (p value) between the competitor algorithm and EMT-ADT, while the Friedman test is used to rank the significance of the compared algorithms statistically. Table 16 shows that EMT-ADT provides higher R + values than MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO and MFMP. The p values of MFEA, MFEARR, AT-MFEA, MTEA-AD, SREMTO and MFMP are less than 0.05, which indicates that the proposed EMT-ADT considerably wins these competitors. For PI + HS, PI + MS and PI + LS problems, the p values of MFDE and TLTLA are greater than 0.05, which indicates that the performance of MFDE and TLTLA is similar to that of EMT-ADT on these problems. For NI + HS, NI + MS and NI + LS problems, the p value of TLTLA is greater than 0.05, which indicates that the performance of TLTLA is similar to that of EMT-ADT on these problems. Furthermore, Table 17 shows the average rank and the overall rank of the compared algorithms for CEC2017 multitask problems. In sum, the average performance of EMT-ADT is first rank on all test problem.

Figure 11 visually shows the ranking of the EMT-ADT and competitor algorithms for CEC2017 multitask problems. The left side demonstrates the ranking of nine algorithms on CEC2017 multitask problems, while the right side illustrates

Fig. 9 Curves of the mean number of the transferred individuals during evolution in EMT-ADT on three test problems. **a** CI-MS, **b** PI-MS, **c** NI-MS





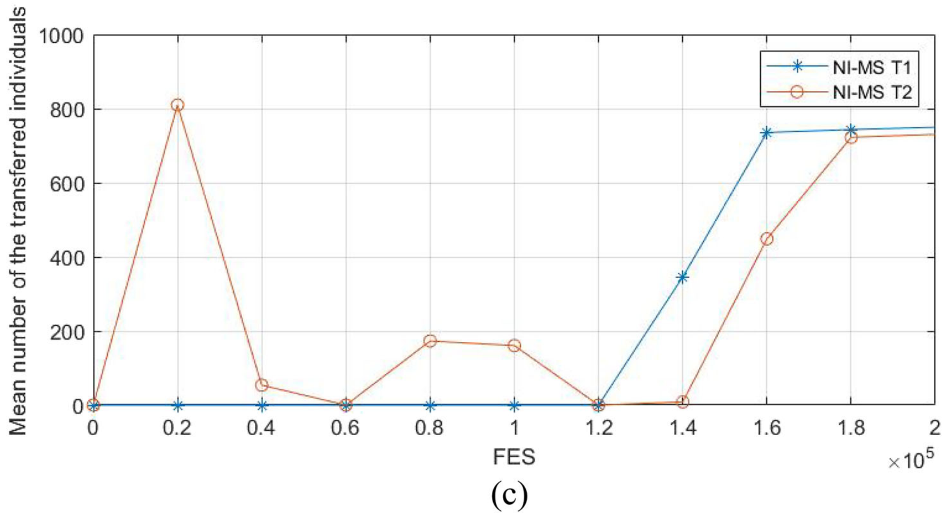


Table 11 Mean fitness values obtained by MFEA, MFEARR, AT-MFEA, MTEA-AD and EMT-ADT on WCCI20-MTSO

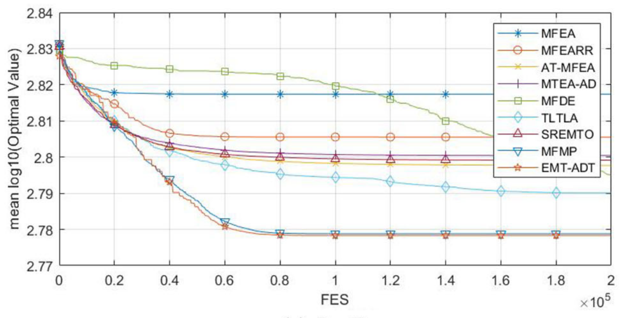
Problem	Task	MFEA	MFEARR	AT-MFEA	MTEA-AD	EMT_ADT
P_1	T_1	6.57E + 02†	6.39E + 02†	6.28E + 02†	6.32E + 02†	**6.00E + 02**
	T_2	6.55E + 02†	6.38E + 02†	6.25E + 02†	6.32E + 02†	**6.00E + 02**
P_2	T_1	7.01E + 02†	7.01E + 02†	7.00E + 02†	7.01E + 02†	**7.00E + 02**
	T_2	7.01E + 02†	7.01E + 02†	7.00E + 02†	7.01E + 02†	**7.00E + 02**
P_3	T_1	1.57E + 06†	1.53E + 06†	9.67E + 05†	1.37E + 06†	**4.97E + 03**
	T_2	1.57E + 06†	3.27E + 06†	1.12E + 06†	1.31E + 06†	**4.72E + 03**
P_4	T_1	1.30E + 03†	1.30E + 03†	1.30E + 03†	1.30E + 03†	**1.30E + 03**
	T_2	1.30E + 03†	1.30E + 03†	1.30E + 03†	1.30E + 03†	**1.30E + 03**
P_5	T_1	1.53E + 03†	1.56E + 03†	1.52E + 03†	1.52E + 03†	**1.51E + 03**
	T_2	1.53E + 03†	1.55E + 03†	1.51E + 03†	1.52E + 03†	**1.51E + 03**
P_6	T_1	1.28E + 06†	1.06E + 06†	7.66E + 05†	7.17E + 05†	**4.85E + 03**
	T_2	7.53E + 05†	1.14E + 06†	4.58E + 05†	6.33E + 05†	**5.90E + 03**
P_7	T_1	2.74E + 03†	3.04E + 03†	2.61E + 03†	2.72E + 03†	**2.25E + 03**
	T_2	2.95E + 03†	3.09E + 03†	2.89E + 03†	2.89E + 03†	**2.38E + 03**
P_8	T_1	**5.20E + 02**–	5.20E + 02–	5.21E + 02†	5.20E + 02–	5.21E + 02
	T_2	5.20E + 02–	5.20E + 02–	5.21E + 02†	**5.20E + 02**–	5.21E + 02
P_9	T_1	7.89E + 03†	8.58E + 03†	6.64E + 03≈	6.91E + 03†	**6.42E + 03**
	T_2	1.62E + 03†	1.62E + 03†	1.62E + 03†	1.62E + 03†	**1.62E + 03**
P_{10}	T_1	1.23E + 04†	2.45E + 04†	1.69E + 04†	1.78E + 04†	**2.20E + 03**
	T_2	1.09E + 06†	5.90E + 05†	1.18E + 06†	9.17E + 05†	**6.62E + 03**
†/≈/–		18/0/2	18/0/2	19/1/0	18/0/2	/

The best solutions are highlighted in bold

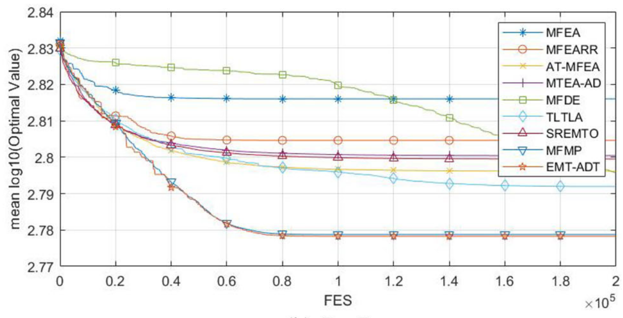
Table 12 Mean fitness values obtained by MFDE, TLTLA, SREMTO, MFMP and EMT-ADT on WCCI20-MTSO

Problem	Task	MFDE	TLTLA	SREMTO	MFMP	EMT_ADT
P_1	T_1	6.24E + 02†	6.17E + 02†	6.30E + 02†	6.01E + 02†	**6.00E + 02**
	T_2	6.25E + 02†	6.19E + 02†	6.30E + 02†	6.01E + 02†	**6.00E + 02**
P_2	T_1	7.00E + 02†	7.01E + 02†	7.01E + 02†	7.00E + 02†	**7.00E + 02**
	T_2	7.00E + 02†	7.01E + 02†	7.01E + 02†	7.00E + 02≈	**7.00E + 02**
P_3	T_1	5.28E + 07†	1.57E + 06†	1.14E + 06†	7.02E + 03†	**4.97E + 03**
	T_2	5.29E + 07†	1.89E + 06†	8.81E + 05†	6.50E + 03†	**4.72E + 03**
P_4	T_1	1.30E + 03†	1.30E + 03†	1.30E + 03†	1.30E + 03†	**1.30E + 03**
	T_2	1.30E + 03†	1.30E + 03†	1.30E + 03†	1.30E + 03†	**1.30E + 03**
P_5	T_1	1.54E + 03†	1.52E + 03†	1.52E + 03†	1.51E + 03†	**1.51E + 03**
	T_2	1.54E + 03†	1.52E + 03†	1.52E + 03†	1.51E + 03†	**1.51E + 03**
P_6	T_1	2.12E + 07†	7.72E + 05†	5.92E + 05†	8.17E + 03†	**4.85E + 03**
	T_2	2.05E + 07†	6.86E + 05†	5.22E + 05†	7.28E + 03†	**5.90E + 03**
P_7	T_1	4.49E + 03†	2.64E + 03†	2.63E + 03†	2.38E + 03†	**2.25E + 03**
	T_2	4.61E + 03†	2.81E + 03†	2.79E + 03†	2.54E + 03†	**2.38E + 03**
P_8	T_1	5.21E + 02†	5.21E + 02†	5.21E + 02†	5.21E + 02†	**5.21E + 02**
	T_2	5.21E + 02†	5.21E + 02†	5.21E + 02†	5.21E + 02†	**5.21E + 02**
P_9	T_1	1.46E + 04†	6.65E + 03≈	**6.37E + 03**≈	7.10E + 03†	6.42E + 03
	T_2	1.62E + 03†	1.62E + 03†	1.62E + 03†	1.62E + 03†	**1.62E + 03**
P_{10}	T_1	7.37E + 04†	2.18E + 04†	1.84E + 04†	2.25E + 03†	**2.20E + 03**
	T_2	2.17E + 07†	1.13E + 06†	6.78E + 05†	1.10E + 04†	**6.62E + 03**
†/≈/–		20/0/0	19/1/0	19/1/0	19/1/0	/

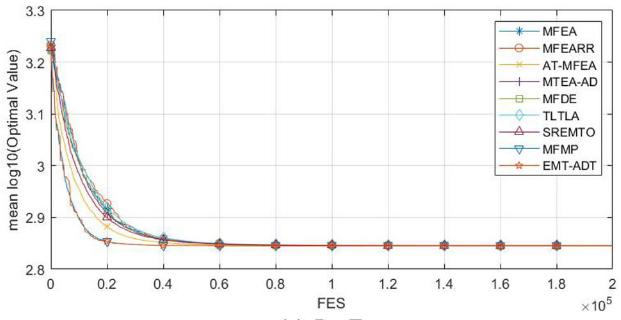
The best solutions are highlighted in bold



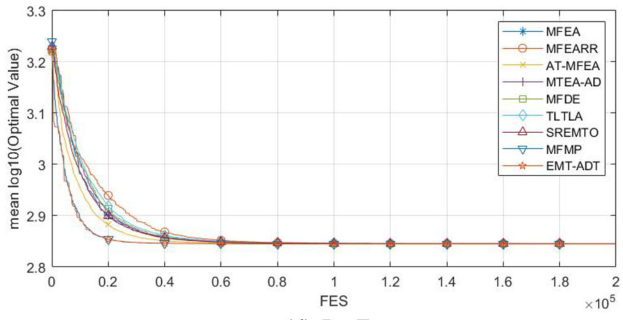
(a) P_1: T_1



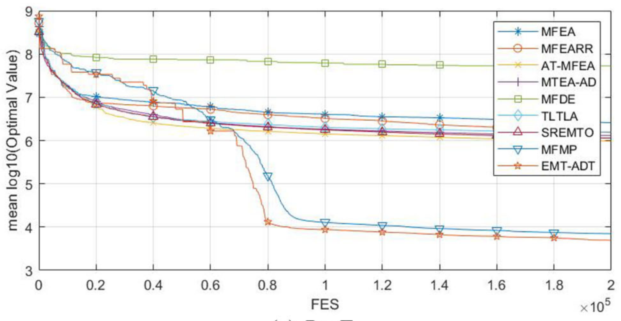
(b) P_1: T_2



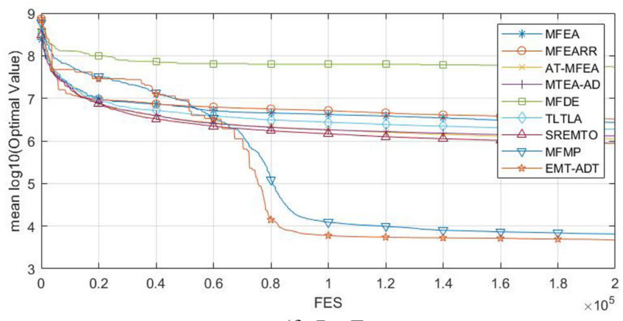
(c) P_2: T_1



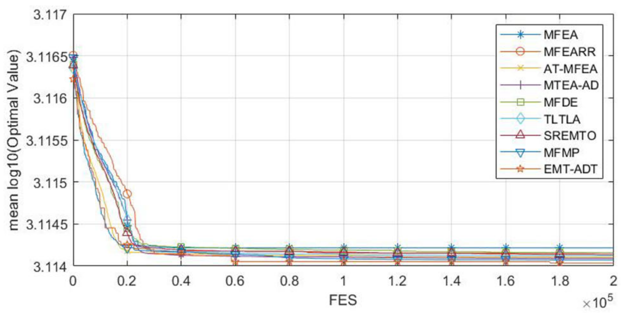
(d) P_2: T_2



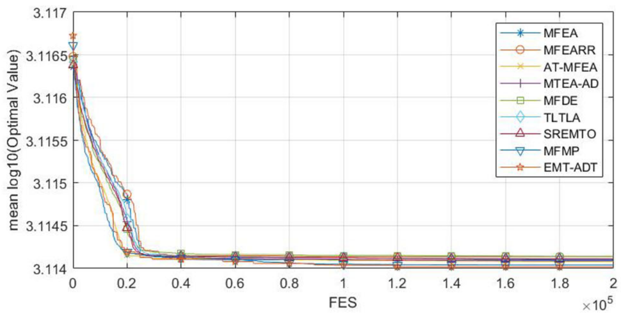
(e) P_3: T_1



(f) P_3: T_2

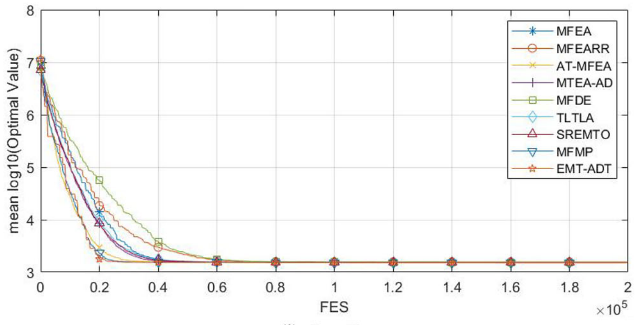


(g) P_4: T_1

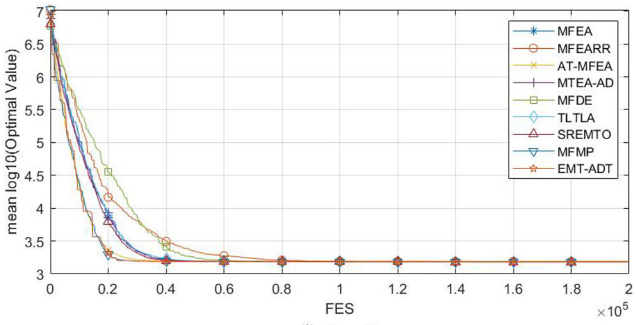


(h) P_4: T_2

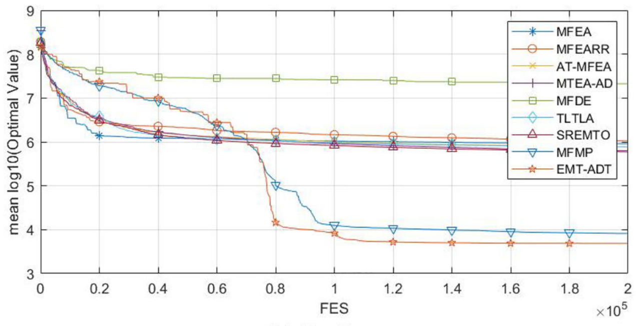
Fig. 10 Convergence curves of the average fitness values obtained by MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO, MFMP and EMT-ADT on WCCI20-MTSO



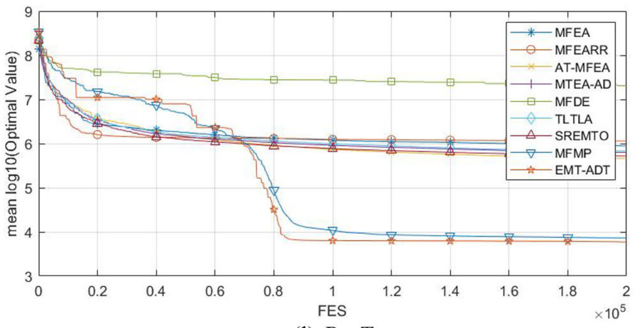
(i) P_5: T_1



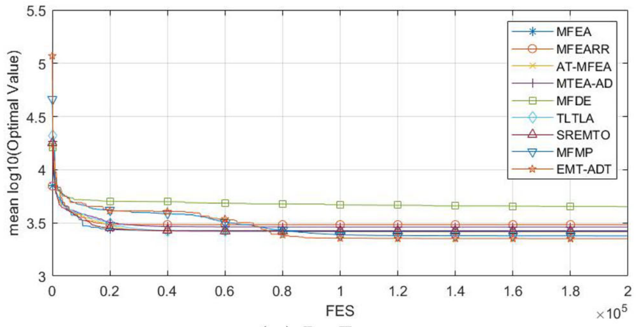
(j) P_5: T_2



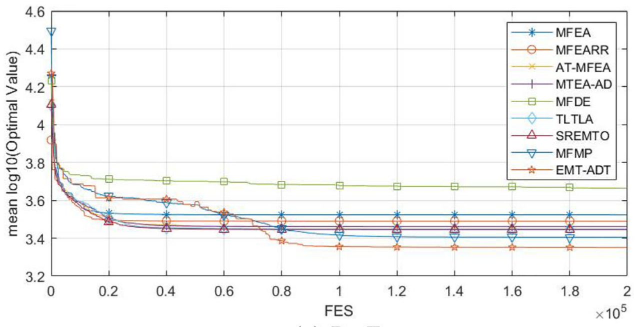
(k) P_6: T_1



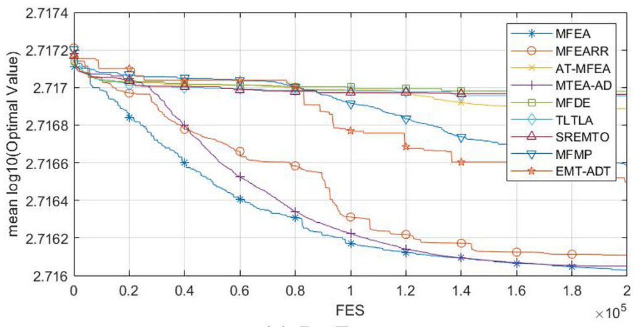
(l) P_6: T_2



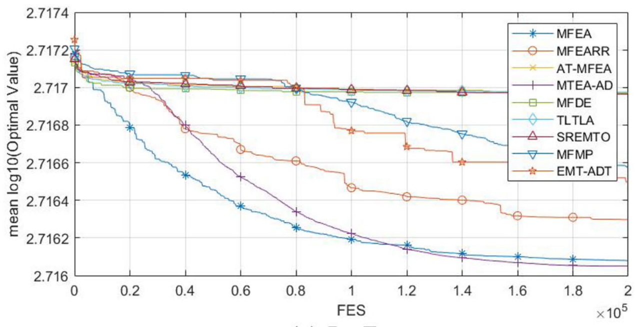
(m) P_7: T_1



(n) P_7: T_2

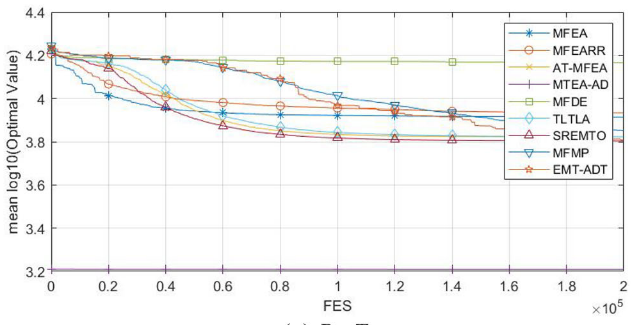


(o) P_8: T_1

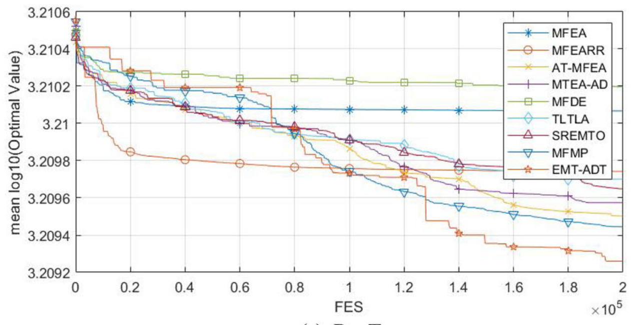


(p) P_8: T_2

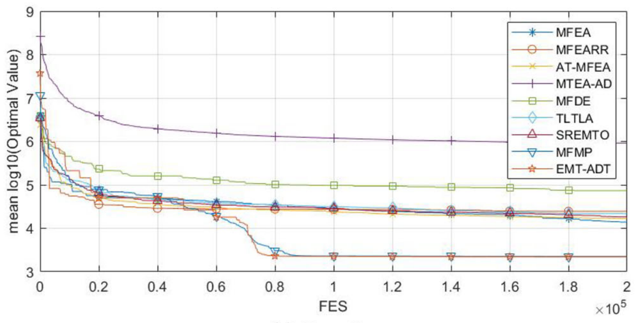
Fig. 10 continued



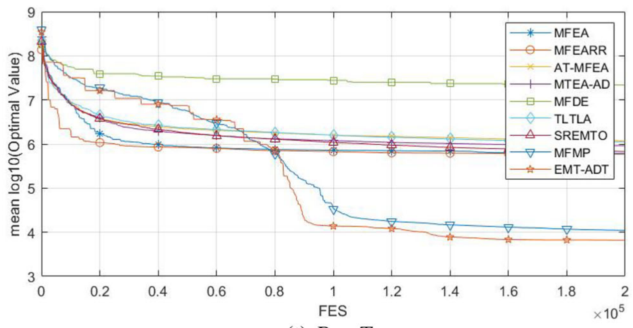
(q) P_9: T_1



(r) P_9: T_2



(s) P_{10}: T_1



(t) P_{10}: T_2

Fig. 10 continued

the bar chart of average ranking of nine algorithms. Since the size of EMT-ADT in radar graph is smaller than that of other compared algorithms, it indicates that the EMT-ADT is competitive. Moreover, EMT-ADT has the shortest bar in the bar chart, which demonstrates that the EMT-ADT is superior to other competitor algorithms.

Population size sensitivity analysis

Population size has a considerable bearing on the rate of convergence. Generally, large population size may slow the convergence rate, while small population may promote a faster convergence [34]. In the proposed algorithm, the parameter γ represents the dispersion degree of population convergence, which is used to control the population size. To analyze the effects of the parameter γ on the performance of EMT-ADT, firstly, the degree of dispersion (DOD) of the population is defined as follows.

$$\mathrm{G} = \frac{\sum_{i=1}^{N} \mathbf{x}_i}{N} \tag{14}$$

$$\mathrm{DOD} = \frac{\sum_{i=1}^{N} \| G - \mathbf{x}_i \|_2}{N} \tag{15}$$

where $\mathbf{x}_i$ is the ith individual in the population P, and N is the population size.

Figure 12 shows the degree of dispersion obtained by EMT-ADT on CEC2017 multitask problems. As seen from Fig. 12, the dispersion of the population is different during the evolution process for different problems with different tasks. To determine the parameter γ for better performance of EMT-ADT, CI + MS, PI + MS and NI + MS problems are selected to test the performance of EMT-ADT with different values of the parameter γ, i.e. $\gamma = 0.0005$, $\gamma = 0.001$, $\gamma = 0.005$, $\gamma = 0.01$, $\gamma = 0.015$, on all tasks.

Figure 13 and Table 18 respectively show the convergence curves and the ranking results on CI + MS, PI + MS and NI + MS problems with different values of the parameter γ. As can be seen in Fig. 13 and Table 18, the algorithm performs better with $\gamma = 0.001$. Therefore, γ is set to 0.001 in the proposed algorithm.

As mentioned before, the population size is adjusted at $\gamma = 0.001$, the algorithm performs the best. Since the dispersion of the population is different during the evolution process for different problems with different tasks, the mean dispersion degree of the population on nine CEC2017 multitask problems is calculated, as seen in Fig. 14. The location of red circle is recorded as (*FES*, γ), which represents the time when the population is adjusted. Therefore, when *FES* is 50,000, that is *FES*/*MAXFES* = cos(γ)/4, the population size should be adjusted to achieve better performance.

Table 13 Mean fitness values obtained by MFEA, MFEARR, AT-MFEA, MTEA-AD, MFDE, TLTLA, SREMTO, MFMP and EMT-ADT on WCCI20-MaTSO

Problem	Task	MFEA	MFEARR	AT-MFEA	MTEA-AD	MFDE	TLTLA	SREMTO	MFMP	EMT-ADT
P_1	T_1	1.19E + 02†	1.59E + 02†	3.06E−01†	2.28E + 02†	2.18E + 01†	1.15E + 02†	4.65E + 02†	8.36E−17†	**8.35E−20**
	T_2	1.10E + 02†	1.52E + 02†	3.22E−01†	2.31E + 02†	1.67E + 01†	1.14E + 02†	4.87E + 02†	4.50E−17†	**2.58E−19**
	T_3	1.29E + 02†	1.74E + 02†	3.38E−01†	2.35E + 02†	1.95E + 01†	1.26E + 02†	4.94E + 02†	1.10E−16†	**6.49E−18**
	T_4	1.49E + 02†	1.85E + 02†	3.69E−01†	2.67E + 02†	1.66E + 01†	1.32E + 02†	5.03E + 02†	2.13E−16†	**2.30E−20**
	T_5	1.29E + 02†	1.59E + 02†	3.58E−01†	2.38E + 02†	1.98E + 01†	1.21E + 02†	4.74E + 02†	8.61E−17†	**1.14E−20**
	T_6	1.26E + 02†	1.76E + 02†	3.10E−01†	2.46E + 02†	2.34E + 01†	1.26E + 02†	4.58E + 02†	7.86E−17†	**1.16E−18**
	T_7	1.32E + 02†	1.59E + 02†	3.48E−01†	2.49E + 02†	2.00E + 01†	1.38E + 02†	5.20E + 02†	6.02E−17†	**1.26E−17**
	T_8	1.37E + 02†	1.75E + 02†	3.18E−01†	2.81E + 02†	2.01E + 01†	1.31E + 02†	5.30E + 02†	2.14E−16†	**4.37E−18**
	T_9	1.10E + 02†	1.62E + 02†	3.10E−01†	2.81E + 02†	1.83E + 01†	1.18E + 02†	5.16E + 02†	6.78E−17†	**4.47E−20**
	T_{10}	1.20E + 02†	1.62E + 02†	3.28E−01†	2.28E + 02†	2.03E + 01†	1.26E + 02†	5.11E + 02†	1.08E−16†	**6.54E−18**
P_2	T_1	8.58E + 04†	5.06E + 04†	3.63E + 02†	9.51E + 04†	2.73E + 04†	3.17E + 04†	3.11E + 05†	4.42E + 01†	**4.10E + 01**
	T_2	5.76E + 04†	3.42E + 04†	2.15E + 02†	8.94E + 04†	1.14E + 04†	1.78E + 04†	2.23E + 05†	8.85E + 01†	**3.95E + 01**
	T_3	9.75E + 04†	4.40E + 04†	1.76E + 02†	1.09E + 05†	1.75E + 04†	3.15E + 04†	3.01E + 05†	5.34E + 01†	**4.28E + 01**
	T_4	9.09E + 04†	3.89E + 04†	2.02E + 02†	1.02E + 05†	2.40E + 04†	2.56E + 04†	2.99E + 05†	4.52E + 01†	**4.24E + 01**
	T_5	7.01E + 04†	4.93E + 04†	3.66E + 02†	8.89E + 04†	1.08E + 04†	1.92E + 04†	2.92E + 05†	6.83E + 01†	**4.33E + 01**
	T_6	1.28E + 05†	4.07E + 04†	2.43E + 02†	1.35E + 05†	1.32E + 04†	2.13E + 04†	2.83E + 05†	4.56E + 01†	**4.34E + 01**
	T_7	6.82E + 04†	5.15E + 04†	8.09E + 01†	1.10E + 05†	1.13E + 04†	2.95E + 04†	2.87E + 05†	4.72E + 01†	**4.41E + 01**
	T_8	1.07E + 05†	4.36E + 04†	1.45E + 02†	1.04E + 05†	1.50E + 04†	2.92E + 04†	3.00E + 05†	5.09E + 01†	**4.16E + 01**
	T_9	6.18E + 04†	4.72E + 04†	1.25E + 03†	8.24E + 04†	8.64E + 03†	2.06E + 04†	2.61E + 05†	4.52E + 01†	**4.30E + 01**
	T_{10}	8.50E + 04†	5.78E + 04†	1.74E + 02†	1.32E + 05†	1.34E + 04†	3.02E + 04†	2.83E + 05†	4.47E + 01†	**3.78E + 01**
P_3	T_1	6.09E + 02†	5.72E + 02†	3.99E + 02†	5.36E + 02†	4.42E + 02†	5.89E + 02†	8.09E + 02†	1.54E + 02†	**1.30E + 02**
	T_2	5.67E + 02†	5.74E + 02†	3.83E + 02†	5.40E + 02†	4.57E + 02†	5.53E + 02†	7.34E + 02†	1.54E + 02†	**1.34E + 02**
	T_3	5.96E + 02†	5.93E + 02†	4.00E + 02†	5.75E + 02†	4.44E + 02†	5.67E + 02†	7.76E + 02†	1.53E + 02†	**1.33E + 02**
	T_4	6.21E + 02†	6.61E + 02†	4.02E + 02†	5.32E + 02†	4.48E + 02†	5.84E + 02†	8.10E + 02†	1.59E + 02†	**1.34E + 02**
	T_5	6.07E + 02†	6.04E + 02†	3.92E + 02†	5.19E + 02†	4.43E + 02†	6.31E + 02†	7.73E + 02†	1.58E + 02†	**1.33E + 02**
	T_6	5.81E + 02†	5.90E + 02†	3.52E + 02†	5.30E + 02†	4.60E + 02†	6.23E + 02†	7.88E + 02†	1.53E + 02†	**1.22E + 02**
	T_7	6.03E + 02†	6.17E + 02†	4.07E + 02†	5.04E + 02†	4.57E + 02†	5.98E + 02†	8.19E + 02†	1.60E + 02†	**1.21E + 02**
	T_8	6.13E + 02†	5.89E + 02†	4.04E + 02†	5.50E + 02†	4.59E + 02†	6.69E + 02†	7.38E + 02†	1.57E + 02†	**1.34E + 02**
	T_9	6.20E + 02†	5.76E + 02†	3.87E + 02†	5.51E + 02†	4.61E + 02†	6.19E + 02†	7.54E + 02†	1.55E + 02†	**1.32E + 02**
	T_{10}	6.22E + 02†	5.91E + 02†	4.17E + 02†	5.50E + 02†	4.57E + 02†	6.45E + 02†	7.67E + 02†	1.48E + 02†	**1.11E + 02**
P_4	T_1	2.09E + 02†	3.24E + 02†	2.06E−01†	7.23E + 02†	2.16E + 01†	1.79E + 02†	5.09E + 02†	1.19E−16†	**3.26E−18**
	T_2	1.49E + 05†	1.36E + 05†	9.91E + 01†	5.45E + 05†	1.45E + 04†	1.14E + 05†	2.48E + 05†	5.17E + 01†	**3.97E + 01**
	T_3	6.31E + 00†	7.19E + 00†	4.70E−01†	8.88E + 00†	3.59E + 00†	6.26E + 00†	8.15E + 00†	2.89E−10†	**1.82E−11**
	T_4	1.75E + 02†	3.26E + 02†	2.86E−01†	7.52E + 02†	1.61E + 01†	1.79E + 02†	5.63E + 02†	3.58E−16†	**1.96E−17**
	T_5	1.65E + 05†	1.55E + 05†	1.04E + 02†	3.45E + 05†	3.53E + 03†	8.74E + 04†	2.28E + 05†	6.22E + 01†	**4.28E + 01**
	T_6	6.30E + 00†	6.96E + 00†	5.30E−01†	8.51E + 00†	3.58E + 00†	5.95E + 00†	8.43E + 00†	2.62E−09†	**3.01E−11**
	T_7	1.80E + 02†	3.33E + 02†	3.69E−01†	7.84E + 02†	1.76E + 01†	1.88E + 02†	5.69E + 02†	1.74E−16†	**2.05E−17**
	T_8	8.88E + 04†	1.55E + 05†	9.12E + 01†	4.12E + 05†	2.72E + 04†	5.20E + 04†	2.92E + 05†	4.49E + 01†	**4.01E + 01**
	T_9	6.30E + 00†	6.97E + 00†	4.29E−01†	8.17E + 00†	3.44E + 00†	5.97E + 00†	8.20E + 00†	8.28E−10†	**2.30E−11**
	T_{10}	1.85E + 02†	3.38E + 02†	3.91E−01†	7.92E + 02†	1.72E + 01†	1.87E + 02†	5.73E + 02†	1.37E−16†	**1.18E−20**
P_5	T_1	6.66E + 02†	7.34E + 02†	4.14E + 02†	6.94E + 02†	4.46E + 02†	6.80E + 02†	8.57E + 02†	1.60E + 02†	**1.33E + 02**
	T_2	1.04E + 00†	1.09E + 00†	5.82E−02†	1.22E + 00†	5.33E−01†	1.04E + 00†	1.15E + 00†	7.81E−13†	**4.66E−15**
	T_3	4.17E + 01†	4.42E + 01†	6.15E + 00†	3.86E + 01†	1.70E + 01†	3.63E + 01†	4.28E + 01†	1.71E + 00†	**1.11E + 00**
	T_4	7.42E + 02†	6.82E + 02†	4.00E + 02†	7.10E + 02†	4.52E + 02†	6.91E + 02†	8.06E + 02†	1.59E + 02†	**1.24E + 02**
	T_5	1.05E + 00†	1.08E + 00†	7.60E−02†	1.23E + 00†	5.33E−01†	1.04E + 00†	1.14E + 00†	7.40E−04†	**5.11E−15**
	T_6	4.53E + 01†	4.42E + 01†	4.50E + 00†	3.77E + 01†	1.77E + 01†	3.57E + 01†	4.07E + 01†	1.55E + 00†	**9.76E−01**
	T_7	7.12E + 02†	7.73E + 02†	4.04E + 02†	7.14E + 02†	4.44E + 02†	7.02E + 02†	8.30E + 02†	1.61E + 02†	**1.43E + 02**
	T_8	1.04E + 00†	1.08E + 00†	9.07E−02†	1.20E + 00†	5.42E−01†	1.04E + 00†	1.15E + 00†	1.27E−12†	**3.11E−15**
	T_9	4.14E + 01†	4.44E + 01†	3.57E + 00†	3.84E + 01†	1.79E + 01†	3.40E + 01†	4.16E + 01†	1.96E + 00†	**1.13E + 00**
	T_{10}	7.31E + 02†	7.02E + 02†	4.11E + 02†	7.07E + 02†	4.49E + 02†	7.09E + 02†	8.05E + 02†	1.72E + 02†	**1.47E + 02**

Table 13 (continued)

Problem	Task	MFEA	MFEARR	AT-MFEA	MTEA-AD	MFDE	TLTLA	SREMTO	MFMP	EMT-ADT
P_6	T_1	9.88E + 04†	1.50E + 05†	1.91E + 02†	6.13E + 05†	1.64E + 04†	6.44E + 04†	2.03E + 05†	4.47E + 01†	**4.22E + 01**
	T_2	1.04E + 00†	1.10E + 00†	4.89E−02†	1.23E + 00†	5.36E−01†	1.05E + 00†	1.14E + 00†	8.90E−06†	**1.82E−14**
	T_3	7.91E + 03–	8.26E + 03–	9.74E + 03†	**6.65E + 03–**	1.33E + 04†	**7.88E + 03–**	**8.14E + 03–**	9.88E + 03†	9.42E + 03
	T_4	1.69E + 05†	3.05E + 05†	2.82E + 02†	6.23E + 05†	4.76E + 04†	1.29E + 05†	3.20E + 05†	5.89E + 01†	**3.45E + 01**
	T_5	1.04E + 00†	1.09E + 00†	5.16E−02†	1.22E + 00†	5.17E−01†	1.05E + 00†	1.15E + 00†	3.20E−12†	**2.66E−15**
	T_6	7.79E + 03–	7.98E + 03–	**6.89E + 03–**	7.33E + 03–	1.35E + 04†	**8.15E + 03–**	**7.77E + 03–**	9.87E + 03†	9.40E + 03
	T_7	1.10E + 05†	1.73E + 05†	2.22E + 02†	6.19E + 05†	6.93E + 03†	1.33E + 05†	3.12E + 05†	4.40E + 01†	**4.10E + 01**
	T_8	1.04E + 00†	1.10E + 00†	7.83E−02†	1.23E + 00†	5.06E−01†	1.05E + 00†	1.14E + 00†	2.82E−12†	**1.67E−15**
	T_9	8.03E + 03–	8.06E + 03–	**6.42E + 03–**	7.44E + 03–	1.30E + 04†	**7.81E + 03–**	8.47E + 03≈	1.00E + 04†	8.76E + 03
	T_{10}	4.87E + 04†	1.39E + 05†	3.82E + 03†	7.58E + 05†	2.53E + 03†	7.75E + 04†	3.76E + 05†	5.52E + 01†	**4.29E + 01**
P_7	T_1	6.72E + 00†	7.23E + 00†	4.25E−01†	8.86E + 00†	3.84E + 00†	5.96E + 00†	9.49E + 00†	6.21E−10†	**1.12E−11**
	T_2	7.05E + 02†	7.47E + 02†	4.05E + 02†	6.91E + 02†	4.60E + 02†	7.40E + 02†	8.30E + 02†	1.66E + 02†	**1.42E + 02**
	T_3	4.39E + 01†	4.74E + 01†	4.03E + 00†	3.79E + 01†	2.04E + 01†	4.11E + 01†	4.57E + 01†	1.91E + 00†	**5.85E−01**
	T_4	6.54E + 00†	7.39E + 00†	3.75E−01†	8.75E + 00†	3.80E + 00†	6.21E + 00†	9.58E + 00†	6.59E−09†	**3.51E−11**
	T_5	7.15E + 02†	6.72E + 02†	4.14E + 02†	6.77E + 02†	4.57E + 02†	6.73E + 02†	7.65E + 02†	1.66E + 02†	**1.45E + 02**
	T_6	4.14E + 01†	4.35E + 01†	3.63E + 00†	3.91E + 01†	1.80E + 01†	3.81E + 01†	4.42E + 01†	2.37E + 00†	**1.13E + 00**
	T_7	6.52E + 00†	7.06E + 00†	3.25E−01†	8.76E + 00†	3.95E + 00†	6.14E + 00†	9.62E + 00†	8.79E−02†	**2.10E−11**
	T_8	7.15E + 02†	7.51E + 02†	4.03E + 02†	6.57E + 02†	4.45E + 02†	6.79E + 02†	7.71E + 02†	1.60E + 02†	**1.19E + 02**
	T_9	4.17E + 01†	4.04E + 01†	4.97E + 00†	3.76E + 01†	1.83E + 01†	3.45E + 01†	4.19E + 01†	1.68E + 00†	**5.20E−01**
	T_{10}	6.53E + 00†	7.25E + 00†	3.47E−01†	8.74E + 00†	3.68E + 00†	6.12E + 00†	9.18E + 00†	2.80E−09†	**1.37E−11**
P_8	T_1	1.07E + 05†	2.65E + 05†	5.98E + 01†	4.98E + 05†	2.25E + 04†	1.30E + 05†	3.99E + 05†	4.49E + 01†	**4.06E + 01**
	T_2	2.03E + 01†	2.04E + 01†	5.55E−01†	1.38E + 01†	1.28E + 01†	2.02E + 01†	2.05E + 01†	7.15E−10†	**5.07E−11**
	T_3	7.53E + 02†	7.89E + 02†	4.06E + 02†	7.38E + 02†	4.44E + 02†	7.64E + 02†	8.15E + 02†	1.60E + 02†	**1.14E + 02**
	T_4	1.05E + 00†	1.12E + 00†	6.07E−02†	1.23E + 00†	4.86E−01†	1.06E + 00†	1.23E + 00†	7.40E−04†	**8.22E−15**
	T_5	5.18E + 01†	5.02E + 01†	5.10E + 00†	4.28E + 01†	3.69E + 01†	4.96E + 01†	5.23E + 01†	4.17E + 00†	**1.93E + 00**
	T_6	1.08E + 05†	3.01E + 05†	2.36E + 02†	8.46E + 05†	1.11E + 04†	8.51E + 04†	8.33E + 05†	5.68E + 01†	**4.29E + 01**
	T_7	2.03E + 01†	2.04E + 01†	5.12E−01†	1.39E + 01†	9.35E + 00†	1.50E + 01†	2.05E + 01†	1.82E−08†	**6.67E−11**
	T_8	7.21E + 02†	7.72E + 02†	3.81E + 02†	7.01E + 02†	4.48E + 02†	7.71E + 02†	9.14E + 02†	1.69E + 02†	**1.46E + 02**
	T_9	1.06E + 00†	1.12E + 00†	5.13E−02†	1.20E + 00†	4.96E−01†	1.06E + 00†	1.23E + 00†	2.01E−12†	**3.55E−15**
	T_{10}	4.98E + 01†	4.81E + 01†	1.12E + 01†	4.04E + 01†	3.65E + 01†	5.00E + 01†	5.58E + 01†	5.03E + 00†	**1.70E + 00**
P_9	T_1	1.01E + 05†	2.54E + 05†	3.22E + 02†	9.15E + 05†	1.52E + 04†	6.92E + 04†	5.73E + 05†	1.34E + 02†	**4.20E + 01**
	T_2	2.03E + 01†	2.04E + 01†	5.13E−01†	1.09E + 01†	8.26E + 00†	1.62E + 01†	2.04E + 01†	4.03E−09†	**3.86E−11**
	T_3	7.24E + 02†	8.00E + 02†	3.66E + 02†	7.31E + 02†	4.50E + 02†	7.65E + 02†	8.26E + 02†	1.76E + 02†	**1.61E + 02**
	T_4	1.06E + 00†	1.11E + 00†	7.90E−02†	1.28E + 00†	4.60E−01†	1.06E + 00†	1.18E + 00†	4.13E−14†	**2.11E−15**
	T_5	5.36E + 01†	5.16E + 01†	3.05E + 01†	4.03E + 01†	3.23E + 01†	5.31E + 01†	5.43E + 01†	3.81E + 00†	**9.24E−01**
	T_6	6.61E + 03–	5.98E + 03–	**3.54E + 03–**	5.80E + 03–	1.27E + 04†	**6.50E + 03–**	**6.21E + 03–**	8.44E + 03†	7.66E + 03
	T_7	1.16E + 05†	3.88E + 05†	2.92E + 02†	8.82E + 05†	1.23E + 04†	2.52E + 05†	7.28E + 05†	5.28E + 01†	**4.21E + 01**
	T_8	2.03E + 01†	1.91E + 01†	4.82E−01†	1.08E + 01†	5.64E + 00†	2.02E + 01†	2.03E + 01†	1.87E−10†	**3.31E−11**
	T_9	7.15E + 02†	8.05E + 02†	3.87E + 02†	7.48E + 02†	4.58E + 02†	6.93E + 02†	8.83E + 02†	1.63E + 02†	**1.44E + 02**
	T_{10}	1.05E + 00†	1.11E + 00†	5.64E−02†	1.30E + 00†	4.62E−01†	1.06E + 00†	1.22E + 00†	4.79E−12†	**1.67E−15**
P_{10}	T_1	2.03E + 01†	2.03E + 01†	3.07E−01†	1.29E + 01†	1.35E + 01†	1.87E + 01†	2.04E + 01†	2.52E−10†	**3.64E−11**
	T_2	7.35E + 02†	8.17E + 02†	2.46E + 02†	7.33E + 02†	4.37E + 02†	7.43E + 02†	7.80E + 02†	1.64E + 02†	**1.44E + 02**
	T_3	1.06E + 00†	1.11E + 00†	5.79E−02†	1.28E + 00†	5.01E−01†	1.06E + 00†	1.24E + 00†	7.40E−04†	**2.66E−15**
	T_4	5.16E + 01†	5.06E + 01†	7.92E + 00†	3.97E + 01†	3.60E + 01†	5.13E + 01†	5.57E + 01†	4.20E + 00†	**1.97E + 00**
	T_5	6.30E + 03–	6.42E + 03–	**4.25E + 03–**	5.80E + 03–	1.22E + 04†	**6.75E + 03–**	**6.76E + 03≈**	8.55E + 03†	7.94E + 03
	T_6	2.03E + 01†	2.04E + 01†	2.34E−01†	1.22E + 01†	1.05E + 01†	2.00E + 01†	2.04E + 01†	5.14E−10†	**1.31E−10**
	T_7	7.42E + 02†	9.04E + 02†	4.10E + 02†	7.63E + 02†	4.44E + 02†	7.43E + 02†	8.01E + 02†	1.74E + 02†	**1.40E + 02**
	T_8	1.06E + 00†	1.11E + 00†	5.04E−02†	1.24E + 00†	4.93E−01†	1.06E + 00†	1.20E + 00†	5.80E−13†	**4.33E−15**
	T_9	5.02E + 01†	5.17E + 01†	1.78E + 01†	4.01E + 01†	3.59E + 01†	4.90E + 01†	5.43E + 01†	4.05E + 00†	**2.35E + 00**
	T_{10}	6.91E + 03–	6.62E + 03–	**5.67E + 03–**	6.14E + 03–	1.24E + 04†	**6.53E + 03–**	**6.79E + 03–**	8.77E + 03†	7.64E + 03
†/≈/–		94/0/6	94/0/6	95/0/5	94/0/6	100/0/0	94/0/6	94/2/4	100/0/0	/

The best solutions are highlighted in bold

Table 14 Mean fitness values obtained by SHADE, LSHADE and EMT-ADT on CEC2017 problems

Problem	Task	EMT-ADT	SHADE	LSHADE
P_1	T_1	**0.00E + 00**	2.99E−13†	0.00E + 00≈
	T_2	**0.00E + 00**	1.59E + 02†	1.16E + 02†
P_2	T_1	**8.88E−16**	4.07E−01†	3.00E−14†
	T_2	**0.00E + 00**	1.58E + 02†	1.11E + 02†
P_3	T_1	**3.36E−03**	2.09E + 01†	2.07E + 01†
	T_2	**6.36E−04**	4.60E + 03†	2.42E + 01†
P_4	T_1	**7.72E + 01**	1.61E + 02†	1.08E + 02†
	T_2	**0.00E + 00**	1.24E−20†	1.13E−27†
P_5	T_1	**7.99E−15**	1.91E−01†	2.65E−14†
	T_2	**4.63E + 00**	5.69E + 01†	3.13E + 01†
P_6	T_1	**8.88E−16**	1.16E−01†	1.58E−14†
	T_2	**1.99E−18**	6.18E−03†	1.99E−18≈
P_7	T_1	**3.88E + 00**	4.77E + 01†	3.87E + 01†
	T_2	**0.00E + 00**	1.58E + 02†	1.11E + 02†
P_8	T_1	5.26E−16	1.74E−13†	**0.00E + 00**–
	T_2	**3.10E−01**	3.17E + 00†	5.66E−01≈
P_9	T_1	**4.87E + 01**	1.62E + 02†	1.14E + 02†
	T_2	**6.36E−04**	4.49E + 03†	3.61E + 01†
†/≈/–		/	18/0/0	14/3/1

The best solutions are highlighted in bold

Table 15 Mean fitness values obtained by SHADE, LSHADE and EMT-ADT on WCCI20_MTSO problems

Problem	Task	EMT-ADT	SHADE	LSHADE
P_1	T_1	**6.00E + 02**	6.03E + 02†	6.00E + 02≈
	T_2	**6.00E + 02**	6.03E + 02†	6.01E + 02≈
P_2	T_1	**7.00E + 02**	7.00E + 02†	7.00E + 02≈
	T_2	**7.00E + 02**	7.00E + 02†	7.00E + 02†
P_3	T_1	**4.97E + 03**	7.95E + 03†	6.10E + 03≈
	T_2	**4.72E + 03**	7.64E + 03†	5.80E + 03†
P_4	T_1	**1.30E + 03**	1.30E + 03†	1.30E + 03†
	T_2	**1.30E + 03**	1.30E + 03†	1.30E + 03†
P_5	T_1	**1.51E + 03**	1.52E + 03†	1.51E + 03†
	T_2	**1.51E + 03**	1.52E + 03†	1.52E + 03†
P_6	T_1	**4.85E + 03**	1.04E + 04†	6.77E + 03†
	T_2	5.90E + 03	7.84E + 03†	**5.85E + 03**≈
P_7	T_1	**2.25E + 03**	2.35E + 03†	2.43E + 03†
	T_2	**2.38E + 03**	2.51E + 03†	2.54E + 03†
P_8	T_1	**5.21E + 02**	5.21E + 02†	5.21E + 02†
	T_2	**5.21E + 02**	5.21E + 02†	5.21E + 02†
P_9	T_1	**6.42E + 03**	9.87E + 03†	8.17E + 03†
	T_2	**1.62E + 03**	1.62E + 03†	1.62E + 03†
P_{10}	T_1	2.20E + 03	2.31E + 03†	**2.18E + 03**–
	T_2	6.62E + 03	8.09E + 03≈	**5.74E + 03**–
†/≈/–		/	19/1/0	13/5/2

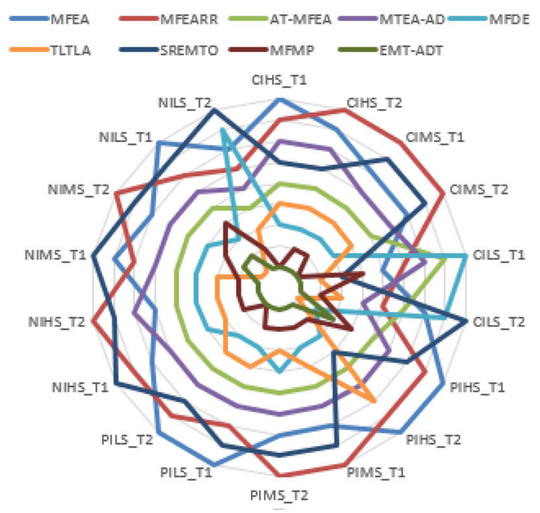
The best solutions are highlighted in bold

Table 16 Wilcoxon test results for EMT-ADT on CEC2017 multitask problems

Functions	CI + HS & MS & LS			PI + HS & MS & LS			NI + HS & MS & LS			All functions		
VS	R^+	R^-	p value	R^+	R^-	p value	R^+	R^-	p value	R^+	R^-	p value
MFEA	21.0	0.0	0.021098	21.0	0.0	0.021098	21.0	0.0	0.021098	171.0	0.0	0.00018
MFEARR	21.0	0.0	0.021098	21.0	0.0	0.021098	21.0	0.0	0.021098	171.0	0.0	0.00018
AT-MFEA	21.0	0.0	0.021098	21.0	0.0	0.021098	21.0	0.0	0.021098	171.0	0.0	0.00018
MTEA-AD	21.0	0.0	0.021098	21.0	0.0	0.021098	21.0	0.0	0.021098	171.0	0.0	0.00018
MFDE	21.0	0.0	0.021098	16.0	5.0	0.208413	21.0	0.0	0.021098	161.0	10.0	0.000934
TLTLA	21.0	0.0	0.021098	15.0	6.0	0.294507	14.0	7.0	0.401678	132.0	39.0	0.040671
SREMTO	21.0	0.0	0.021098	21.0	0.0	0.021098	21.0	0.0	0.021098	171.0	10.0	0.00018
MFMP	19.5	1.5	0.046399	15.0	0.0	0.030971	21.0	0.0	0.021098	151.5	1.5	0.000352

Table 17 Average ranking of the algorithms (Friedman)

Functions	CI + HS & MS & LS		PI + HS & MS & LS		NI + HS & MS & LS		All functions	
Algorithm	Ave. rank	Overall rank	Ave. rank	Overall rank	Ave. rank	Overall rank	Ave. rank	Overall rank
MFEA	7.25	8	8.3333	9	7.3333	7	7.6389	8
MFEARR	7.5833	9	8.1667	8	7.6667	8	7.8056	9
AT-MFEA	5.6667	5	5	5	4.8333	5	5.1667	5
MTEA-AD	6.1667	6	6	6	6	6	6.0556	6
MFDE	4.5	4	2.8333	3	4.5	4	3.9444	4
TLTLA	3.5	3	3.8333	4	2.3333	2	3.2222	3
SREMTO	6.6667	7	7	7	8.5	9	7.3889	7
MFMP	2.1667	2	2.0833	2	2.3333	2	2.1944	2
EMT-ADT	1.5	1	1.75	1	1.5	1	1.5833	1



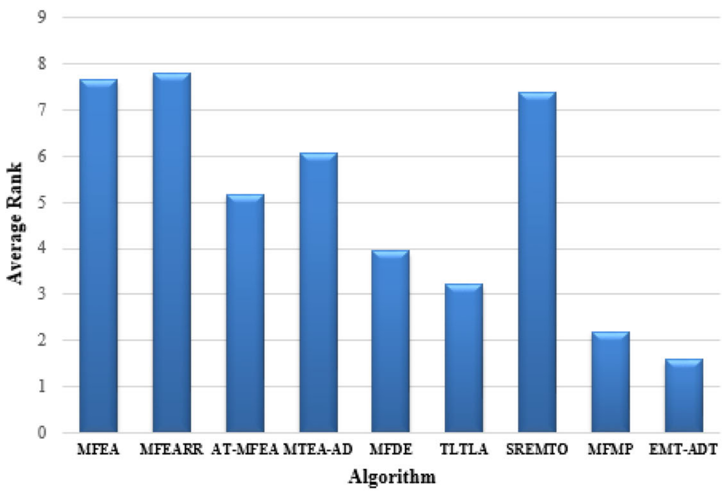


Fig. 11 The radar graph and bar charts of different algorithms on CEC2017 multitask problems

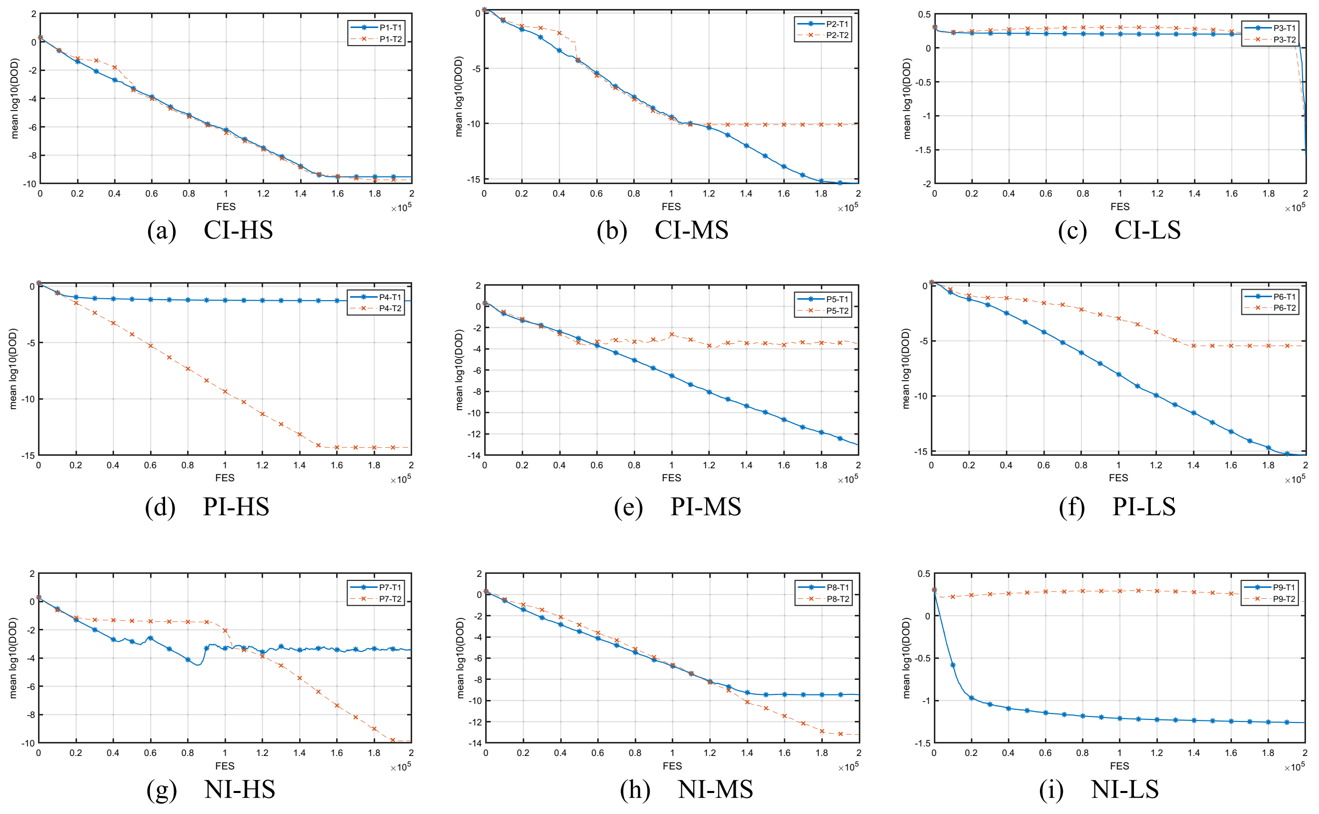


Fig. 12 The degree of dispersion obtained by EMT-ADT on CEC2017 multitask problems

Results on real-world problems

In this section, two real-world problems called the traveling salesman problem (TSP) [35] and traveling repairman problem (TRP) [36] are used to verify the practicability of the proposed EMT-ADT algorithm. More specifically, TSP and TRP are constructed as a multitask optimization problem, which is asked for a tour with minimum cost.

Given a list of n cities and a list of maintenance times for n cities, TSP aims to minimize the total time to visit these cities, while TRP aims to minimize the sum of the elapsed times for all customers that have to wait before being served [37]. Each city is allowed to visit only once, and finally returns to the origin city. Given a distance matrix $C = \{c(x_i, x_j)|i, j = 1, 2, \ldots, n\}$, where $c(x_i, x_j)$ is the distance between two cities x_i and x_j. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ to be a tour, i.e. a solution of the objective function. $P(x_1, x_n)$ represents the path from x_1 to x_n. For the TSP problem, the total time $ls(p(x_1, x_n))$ from starting city x_1 to x_n is computed as

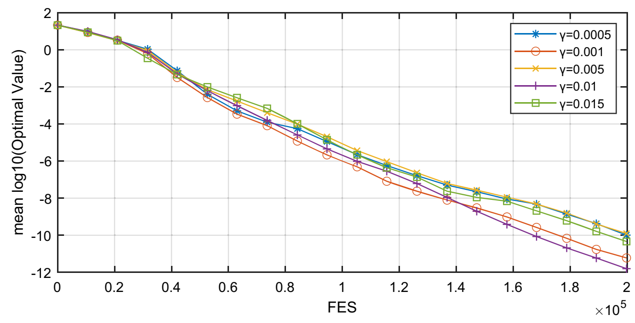
$$ls(p(x_1, x_n)) = \sum_{i=1}^{n-1} c(x_i, x_{i+1}) + c(x_1, x_n) \tag{16}$$

For the TRP problem, the total time $lr(p(x_1, x_n))$ from starting city x_1 to x_n is computed as

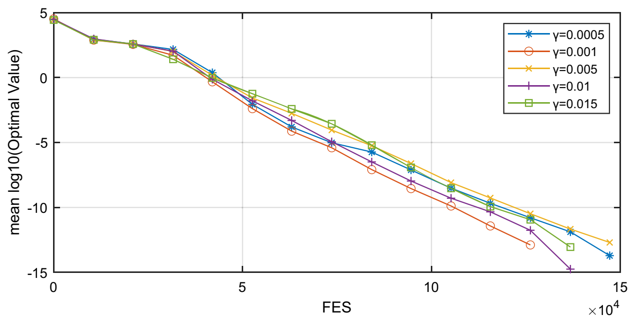
$$lr(p(x_1, x_n)) = \sum_{i=1}^{n-1}\left(c(x_i, x_{i+1}) + \sum\nolimits_{j=1}^{i} r_j\right) + c(x_1, x_n) + r_n \tag{17}$$

Generally, a unified representation scheme is used in multitask optimization algorithm, in which every solution is encoded by a random key between 0 and 1. Since the paths in TSP and TRP problems should be a set of integers without duplication, the real number encoding is converted to permutation number encoding. Specifically, all dimension values of the solution $\mathbf{x}$ are sorted in ascending order. Afterwards, the permutation solution $\mathbf{x}'$ of solution $\mathbf{x}$ is obtained according to the ranking of dimension values. Figure 15 shows a simple example. For a solution $\mathbf{x} = (0.21, 0.34, 0.84, 0.67, 0.11, 0.08, 0.55)$, its permutation solution $\mathbf{x}' = (3, 4, 7, 6, 2, 1, 5)$ denotes that the traveler starts from the third city, visits the fourth, seventh, sixth, second, first and fifth cities, and then returns to the third city.

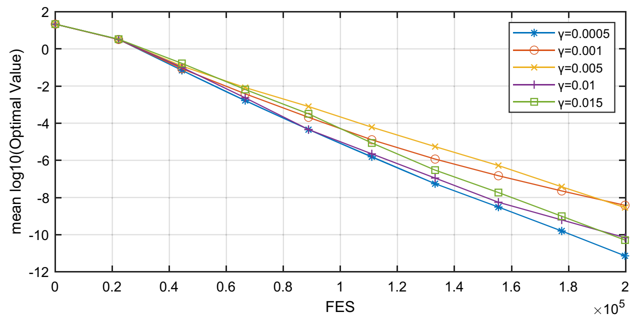
In this experiment, we investigate the EMT-ADT, MFDE, MFPSO and the MFMP. Ten groups of problems are randomly selected from TSPLIB [38]. Suppose travelers travel between cities at a speed of 60 km/h. Each group is composed of a TSP test case and a TRP test case. For each test



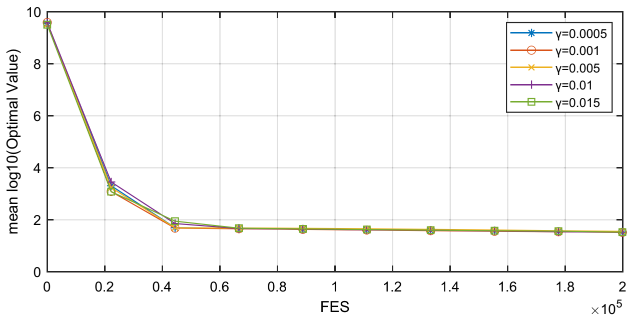
(a) CI-MS: T_1



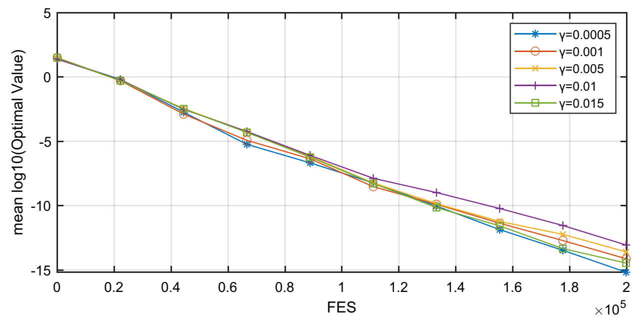
(b) CI-MS: T_2



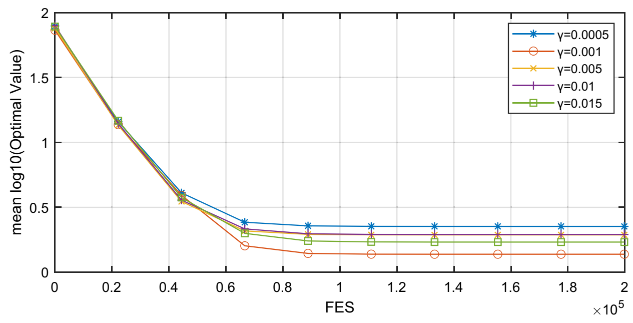
(c) PI-MS: T_1



(d) PI-MS: T_2



(e) NI-MS: T_1



(f) NI-MS: T_2

Fig. 13 Convergence curves with different γ

Table 18 Ranking of different γ on CI + MS, PI + MS and NI + MS problems

γ	CI + MS		PI + MS		NI + MS		Ave. rank
	T_1	T_2	T_1	T_2	T_1	T_2	
$\gamma = 0.0005$	4	4	1	4	1	5	3.17
$\gamma = 0.001$	2	1	5	1	3	1	2.17
$\gamma = 0.005$	5	5	4	5	4	4	4.50
$\gamma = 0.01$	1	2	3	2	5	3	2.67
$\gamma = 0.015$	3	3	2	3	2	2	2.50

Table 19 Average time (hours) spent by the traveler obtained by each algorithm

Instances	Task	EMT_ADT	MFDE	MFPSO	MFMP
berlin52	TSP	**159.3643486**	229.1462823	283.4451226	312.347257
	TRP	**8656.916011**	9869.060957	11,966.12081	10,557.00903
ch130	TSP	**498.7551751**	508.8042919	670.930163	614.6604992
	TRP	**62,197.5479**	66,509.4319	83,890.50617	73,100.84403
eil51	TSP	**9.862641757**	11.74736244	18.21326348	16.6438674
	TRP	**5358.397389**	5482.648334	6088.072607	5412.303255
eil76	TSP	20.46134622	**17.0673391**	34.26721908	32.03857071
	TRP	**11,206.43997**	11,596.42367	13,366.81866	11,439.34994
eil101	TSP	**25.56555029**	25.95953947	37.96147829	44.53472604
	TRP	**19,772.36955**	20,877.96223	25,110.20341	19,964.23559
kroA100	TSP	**956.0707501**	984.8757903	2435.276189	1294.539043
	TRP	**62,854.12605**	67,639.09777	139,053.2715	104,441.4754
kroB100	TSP	**832.8762773**	952.0612095	2317.943192	2060.514439
	TRP	**62,024.19728**	67,160.81999	134,719.4472	113,395.1831
pr76	TSP	**2832.830837**	3430.747388	5108.208294	7206.661393
	TRP	**103,938.6386**	123,705.6655	201,965.7336	215,881.814
rat99	TSP	59.56985418	61.30108744	**41.23388859**	102.4018896
	TRP	**21,927.70112**	23,856.41117	24,195.14046	22,098.80504
rd100	TSP	**269.5498536**	383.716592	769.9456139	703.420969
	TRP	**34,068.40246**	41,033.41973	61,720.5967	52,389.24794

The best solutions are highlighted in bold

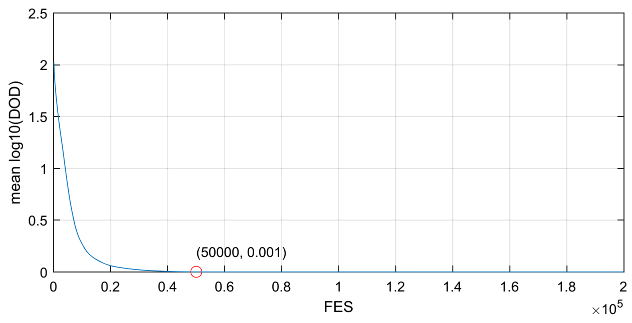


Fig. 14 Mean dispersion degree obtained by EMT-ADT on CEC2017 multitask problems

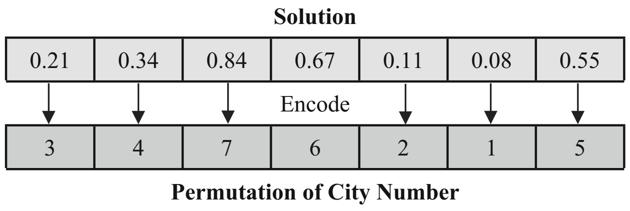


Fig. 15 An example of mapping a solution to a permutation of city number

case, the average time spent by the traveler obtained by each algorithm on 30 independent runs is presented in Table 19. The number following the instance name indicates the number of city nodes. Table 19 shows that EMT-ADT achieves a tour with minimum cost against all peer competitors on almost all the instances. EMT-ADT wins 18 out of the 20 competitions, which demonstrates the superior performance of the proposed EMT-ADT. Figures 16 and 17 show the optimal TSP tour and the optimal TRP tour obtained by different algorithms on the eil51 instance, respectively. Experimental results show that EMT-ADT obtains better solutions than three state-of-the-art multitasking optimization algorithms.

Conclusions

To enhance the positive knowledge transfer between tasks, this paper proposed an evolutionary multitasking optimization algorithm with adaptive transfer strategy based on the decision tree (EMT-ADT). A method of quantifying the transfer ability of individuals is proposed to select individuals with high transfer ability. A decision tree is constructed to predict the transfer ability of individuals in the archive. Then, individuals with high transfer ability are selected to conduct knowledge transfer, which can improve the performance of the algorithm. The effectiveness of EMT-ADT is verified through a comparison with eight state-of-the-art algorithm, i.e. MFEA, MFEARR, AT-MFEA, MTEA-AD,

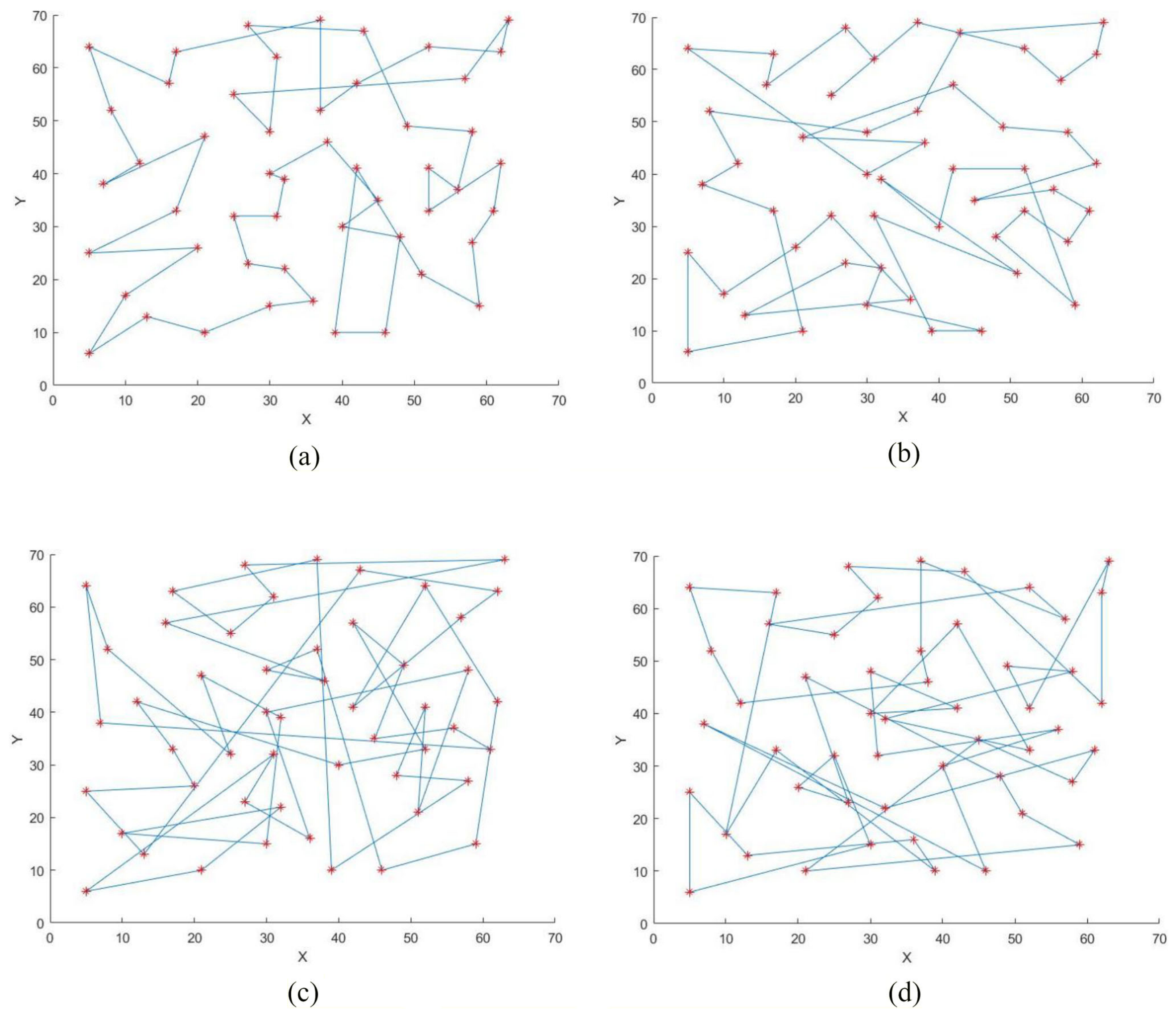


Fig. 16 The optimal TSP tour on the eil51 instance obtained by different algorithms. **a** EMT-ADT, **b** MFDE, **c** MFPSO, **d** MFMP

MFDE, TLTLA, SREMTO and MFMP. The experimental results indicated that EMT-ADT performed better on most CEC2017, WCCI20-MTSO and WCCI20-MaTSO multitask problems.

As mentioned in [39], several specific actions are advocated to bright light to the field of evolutionary multitask optimization. For example, the computational complexity of evolutionary multitasking methods should be regarded as a metric for the evaluation of algorithms. Future work will focus on improving the effectiveness of knowledge transfer operations between tasks with low correlation, and reducing the computational complexity for multifactorial problems with more than two tasks. Moreover, EMT-ADT will be used to solve multi-objective multitask optimization problems.

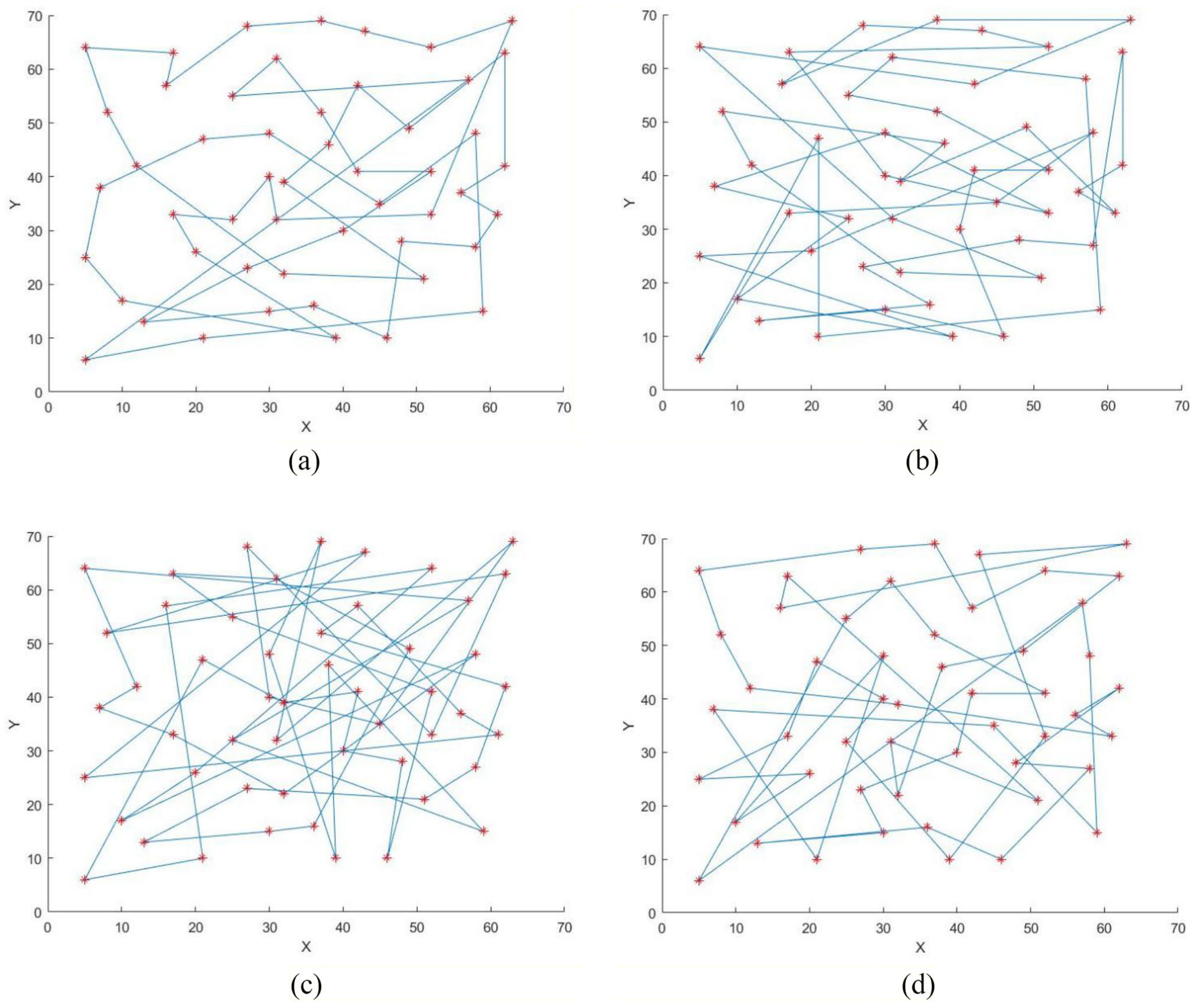


Fig. 17 The optimal TRP tour on the eil51 instance obtained by different algorithms. **a** EMT-ADT, **b** MFDE, **c** MFPSO, **d** MFMP

Acknowledgements This research is partly supported by the National Natural Science Foundation of China under Project Code (62176146), Special project of Education Department of Shaanxi Provincial Government for Local Services (21JC026).

Data availability The experimental data used to support the findings of this study are included within the article.

Declarations

Conflict of interest The authors declare that they have no conflict of interest. The authors have no relevant financial or non-financial interests to disclose.

References

1. Gupta A, Ong YS, Feng L (2016) Multifactorial evolution: toward evolutionary multitasking. IEEE Trans Evolut Comput 20:343–357
2. Zhang F, Mei Y, Nguyen S, Zhang M, Tan KC (2021) Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. IEEE Trans Evolut Comput 25:651–665

3. Binh HTT, Thang TB, Thai ND, Thanh PD (2021) A bi-level encoding scheme for the clustered shortest-path tree problem in multifactorial optimization. Eng Appl Artif Intell 100:104187
4. Zhang B, Qin AK, Sellis T (2018) Evolutionary feature subspaces generation for ensemble classification. In: 2018 genetic and evolutionary computation conference (GECCO), Japan, pp 577–584
5. Xu Z, Liu X, Zhang K, He J (2022) Cultural transmission based multi-objective evolution strategy for evolutionary multitasking. Inf Sci 582:215–242
6. Bali KK, Gupta A, Feng L, Ong YS, Siew TP (2017) Linearized domain adaptation in evolutionary multitasking. In: 2017 IEEE congress on evolutionary computation (CEC), Donostia, pp 1295–1302
7. Feng L, Zhou L, Zhong J, Gupta A, Ong YS, Tan KC, Qin AK (2019) Evolutionary multitasking via explicit autoencoding. IEEE Trans Cybern 49:3457–3470
8. Xue X, Zhang K, Tan KC, Feng L, Wang J, Chen G, Zhao X, Zhang L, Yao J (2022) Affine transformation-enhanced multifactorial optimization for heterogeneous problems. IEEE Trans Cybern 52:6217–6231
9. Bali KK, Ong Y-S, Gupta A, Tan PS (2020) Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II. IEEE Trans Evolut Comput 24:69–83
10. Li G, Lin Q, Gao W (2020) Multifactorial optimization via explicit multipopulation evolutionary framework. Inf Sci 512:1555–1570
11. Da B, Gupta A, Ong YS (2019) Curbing negative influences online for seamless transfer evolutionary optimization. IEEE Trans Cybern 49:4365–4378
12. Gao F, Gao W, Huang L, Xie J, Gong M (2022) An effective knowledge transfer method based on semi-supervised learning for evolutionary optimization. Inf Sci 612:1127–1144
13. Zheng X, Qin AK, Gong M, Zhou D (2020) Self-regulated evolutionary multitask optimization. IEEE Trans Evolut Comput 24:16–28
14. Cai Y, Peng D, Liu P, Guo J (2021) Evolutionary multi-task optimization with hybrid knowledge transfer strategy. Inf Sci 580:874–896
15. Liang Z, Liang W, Wang Z, Ma X, Liu L, Zhu Z (2022) Multiobjective evolutionary multitasking with two-stage adaptive knowledge transfer based on population distribution. IEEE Trans Syst Man Cybern Syst 52:4457–4469
16. Ding J, Yang C, Jin Y, Chai T (2019) Generalized multitasking for evolutionary optimization of expensive problems. IEEE Trans Evolut Comput 23:44–58
17. Safavian SR, Landgrebe D (1991) A survey of decision wee classifier methodology. IEEE Trans Syst Man Cbybern Syst 21:660–674
18. Moral-García S, Abellán J, Coolen-Maturi T, Coolen FPA (2022) A cost-sensitive imprecise credal decision tree based on nonparametric predictive inference. APPL Soft Comput 123(108916):1–14
19. Segatori A, Marcelloni F, Pedrycz W (2018) On distributed fuzzy decision trees for big data. IEEE Trans Fuzzy Syst 26:174–192
20. Lin J, Liu HL, Tan KC, Gu F (2021) An effective knowledge transfer approach for multiobjective multitasking optimization. IEEE Trans Cybern 51:3238–3248
21. Gupta A, Ong YS, Feng L, Tan KC (2017) Multiobjective multifactorial optimization in evolutionary multitasking. IEEE Trans Cybern 47:1652–1665
22. Wu D, Tan X (2020) Multitasking genetic algorithm (MTGA) for fuzzy system optimization. IEEE Trans Fuzzy Syst 28:1050–1061
23. Feng L, Zhou W, Zhou L, Jiang SW, Zhong JH, Da BS, Zhu ZX, Wang Y (2017) An empirical study of multifactorial PSO and multifactorial DE. In: 2017 IEEE congress on evolutionary computation (CEC), Donostia, pp 921–928
24. Li W, Lei Z, Yuan J, Luo H, Xu Q (2021) Enhancing the competitive swarm optimizer with covariance matrix adaptation for large scale optimization. Appl Intell 51:4984–5006
25. Zheng X, Lei Y, Gong M, Tang Z (2016) Multifactorial brain storm optimization algorithm. In: International conference on bio-inspired computing: theories and applications, pp 47–53
26. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: 2013 IEEE congress on evolutionary computation, Cancun, pp 71–78
27. Da BS, Ong YS, Feng L, Qin AK, Gupta A, Zhu ZX, Ting CK, Tang K, Yao X (2016) Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metrics and baseline results. Technical Report, Nanyang Technological University
28. Feng L, Qin K, Gupta A, Yuan Y, Ong Y, Chi X (2019) IEEE CEC 2019 competition on evolutionary multi-task optimization
29. Wen YW, Ting CK (2017) Parting ways and reallocating resources in evolutionary multitasking. In: 2017 IEEE congress on evolutionary computation (CEC), Donostia, pp 2404–2411
30. Ma X, Chen Q, Yu Y, Sun Y, Ma L, Zhu Z (2020) A two-level transfer learning algorithm for evolutionary multitasking. Front Neurosci 13:1408
31. Wang C, Liu J, Wu K, Wu Z (2022) Solving multitask optimization problems with adaptive knowledge transfer via anomaly detection. IEEE Trans Evolut Comput 26:304–318
32. Wang Y, Cai ZX, Zhang QF (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evolut Comput 15:55–66
33. Alcalá-Fdez J, Sánchez L, García S, del Jesus MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM, Fernández JC, Herrera F (2009) KEEL: a software tool to assess evolutionary algorithms to data mining problems. Soft Comput 13:307–318
34. Tanabe R, Fukunaga AS (2014) Improving the search performance of SHADE using linear population size reduction. In: 2014 IEEE congress on evolutionary computation (CEC), Beijing, pp 1658–1665
35. Lawler EL, Lenstra JK, Kan AR, Shmoys DB (1985) The traveling salesman problem: a guided tour of combinatorial optimization, vol 3. Wiley, New York
36. Silva MM, Subramanian A, Vidal T, Ochi LS (2012) A simple and effective metaheuristic for the minimum latency problem. Eur J Oper Res 221(3):513–520
37. Ban HB, Pham DH (2022) Multifactorial evolutionary algorithm for simultaneous solution of TSP and TRP. Comput Inform 40:1370–1397
38. Reinelt G (1995) Tsplib95. Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), vol 338, Heidelberg, pp 1–16
39. Osaba E, Del Ser J, Suganthan PN (2022) Evolutionary multitask optimization: fundamental research questions, practices, and directions for the future. Swarm Evol Comput 75(101203):1–9